Computer persönlich

Ein Markt & Technik Buch

# Das Apple IIc Buch

Franz Santjohanser

Alles über Hardware, Integer – BASIC, Applesoft – BASIC, Maschinensprache, DOS, ProDOS und die Maus.

Das Apple IIc-Buch

Franz Santjohanser

# Das Apple IIc-Buch

Alles über Hardware,
Integer-BASIC,
Applesoft-BASIC,
Maschinensprache,
DOS, ProDOS und
die Maus.

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

**Santjohanser, Franz:**
[Das Apple-zwei-c-Buch]
Das Apple-IIc-Buch : alles über Hardware, Integer-BASIC, Applesoft-BASIC, Maschinensprache, DOS, ProDOS u. d. Maus / Franz Santjohanser. — Haar bei München : Markt-und-Technik-Verlag, 1985.—
(Computer persönlich)
ISBN 3-89090-078-X

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

Apple IIc® ist ein Warenzeichen der Apple Computer, Inc., USA

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
89 88 87 86 85

ISBN 3-89090-078-X

© 1985 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch, Günzburg
Printed in Germany

## Vorwort

Das Angebot an Heim- und Personalcomputern ist heute so vielseitig, daß eine Orientierung nicht nur dem Außenstehenden unmöglich ist, sondern auch dem Eingeweihten oft genug schwerfällt. Die Firma Apple Computer Inc. hat in diesem Markt oft eine Vorreiterstellung eingenommen. Mit Computern wie Apple II, LISA und Macintosh wurden Meilensteine in der Computergeschichte gesetzt.

Auch mit dem Apple IIc steht Ihnen ein Gerät zur Verfügung, das sowohl dem Anfänger als auch dem Profi viele Möglichkeiten bietet, einfach und doch effektiv zu arbeiten. Das APPLE-II-c-BUCH versteht sich als Ratgeber, geschrieben aus praktischer Erfahrung mit dem Apple IIc und seinen Peripheriegeräten. Es soll Ihnen das Rüstzeug geben, Ihren Computer optimal zu nutzen und sinnvoll auf- und auszubauen. So führt es Sie in leichtverständlicher Weise mit praktischen Beispielen in die Programmiersprache BASIC ein und hilft Ihnen, sich mit den systembezogenen Eigenschaften des Computers vertraut zu machen. Es sollen aber auch die Grenzen des Gerätes gezeigt werden.

Ob Sie nun den Apple IIc als Spielepartner oder als ernsthaftes Arbeitswerkzeug benutzen wollen, bleibt Ihnen überlassen. All diese Möglichkeiten werden beschrieben und sollen Ihnen helfen, sich mit dem Apple IIc vertraut zu machen. Das Buch wird Sie aber auch dann nicht allein lassen, wenn Sie Ihren Computer benutzen wie die sonstigen Dinge des täglichen Lebens. Als Nachschlagwerk dient es der Information, die zeigt, was der Computer wirklich ist: ein Werkzeug, um Ihnen die Freizeit zu verschönern und die Arbeit ein wenig leichter zu machen.

Franz Santjohanser

## Inhaltsverzeichnis


**1 Die Apple-Geschichte.....15**

**2 Das Apple-IIc-System.....23**

2.1 Erster Kontakt mit dem Computer.....26
2.1.1 Die Tastatur.....26
2.1.2 Schalter und Taster.....27
2.1.3 Die Kontrollampen.....28
2.1.4 Lautstärkeregler, Kopfhöreranschluß.....28
2.1.5 Das eingebaute Diskettenlaufwerk.....29
2.1.6 Der Handgriff.....29
2.1.7 Joystick- und Maus-Anschluß.....31
2.1.8 Serieller Anschluß 2.....32
2.1.9 Videosignalbuchsen.....32
2.1.10 Diskettenlaufwerk-Anschluß.....33
2.1.11 Serieller Anschluß 1.....33
2.1.12 Netzanschluß.....33
2.1.13 Netzteil/Transformator.....33
2.1.14 Die Lüftungsschlitze.....34
2.1.15 Die Unterseite.....34
2.1.16 Der PAL-Modulator/Adapter.....35
2.1.17 Die Handbücher.....35
2.1.18 Die System- und Einführungsdisketten.....36

2.2 Das Innenleben des Apple IIc.....41
2.2.1 Der Mikroprozessor (CPU).....42
2.2.2 Die Speicherbausteine.....44
2.2.3 Das Diskettenlaufwerk.....44
2.2.4 Die Tastatur.....46
2.2.5 Bildschirm, Monitor.....46
2.2.6 Der Monitor.....47
2.2.7 Das Fernsehgerät.....48
2.2.8 Der LCD-Bildschirm.....49

2.3 Peripheriegeräte.....50
2.3.1 Drucker.....51
2.3.2 Plotter.....51
2.3.3 Modem und Akustikkoppler.....53
2.3.4 Joystick und Maus.....54
2.3.5 Das externe Diskettenlaufwerk.....54

2.3.6 Hard Disk/Festplattenlaufwerk.....55
2.3.7 Btx-Modulbox.....55
2.3.8 Sprachgenerator.....56
2.3.9 Akku, Stromversorgung, Transporttasche.....56

**3 Die Bedienung des Apple IIc.....57**

3.1 Aufstellung und Inbetriebnahme.....59
3.2 Verbindung mit dem Netz.....60
3.3 Verbindung mit dem Bildschirm.....62
3.4 Einschalten, Abstimmung mit dem Bildschirm.....63
3.5 Systemmeldung.....63
3.6 Wenn nichts passiert!.....64
3.7 Der Cursor.....64
3.8 Bildschirmformat.....65
3.9 Die Tastatur.....66
3.10 Kompatibilität zu anderen Apple-Computern.....68

**4 BASIC auf dem Apple IIc.....69**

4.1 BASIC-Startdiskette unter ProDOS.....73
4.2 BASIC-Startdiskette unter DOS 3.3.....78
4.3 Integer-BASIC.....79
4.4 Applesoft-BASIC.....80
4.5 Programmieren in BASIC.....80
4.6 Umschalten von Applesoft- auf Integer-BASIC.....82
4.7 Was bei der Tastatur zu beachten ist.....82

4.8 BASIC-Betriebsarten.....84
4.8.1 Die Direktanweisung.....85
4.8.2 Programmieren.....91

4.9 Die Programmzeilennumerierung.....94
4.10 Anzeigen der Programmzeilen.....95
4.11 Anmerkungen in Programmzeilen.....97

4.12 Editieren des Programms.....98
4.12.1 Zeichen einfügen und verändern.....99
4.12.2 Löschen einzelner Zeichen/ganzer Zeilen.....100
4.12.3 Bildschirm löschen.....102
4.12.4 Programmzeilen einfügen.....102

4.13 Zahlen, Texte und Variable.....103
4.13.1 Ganze Zahlen.....104
4.13.2 Dezimalzahlen.....104
4.13.3 Texte.....105
4.13.4 Variable.....105
4.13.5 Numerische Variable.....106
4.13.6 Textvariable.....109
4.13.7 Ganzzahl-Variable.....109

4.14 Der Variablenaufbau in INTEGER-BASIC.....110
4.15 Der Variablenaufbau in APPLESOFT-BASIC.....110
4.16 Reservierte Worte.....111
4.17 Operatoren.....111
4.18 Vergleichende Operatoren.....113
4.19 Boolesche Algebra.....114
4.20 Tabellen.....114
4.21 DATA und READ.....118
4.22 Programmaufbau und Ablauf.....120
4.23 Sprunganweisung.....120
4.24 Programmschleifen.....123
4.25 Das Unterprogramm.....124
4.26 IF-THEN-Anweisung.....128

4.27 Ein- und Ausgabefunktionen.....129
4.27.1 Das Komma in einer PRINT-Anweisung.....130
4.27.2 Das Semikolon in einer PRINT-Anweisung.....131
4.27.3 Tabulatoren setzen.....132
4.27.4 TAB.....133
4.27.5 HTAB.....133
4.27.6 VTAB.....134
4.27.7 SPC.....134
4.27.8 POS.....135

4.28 Darstellungsvarianten.....135
4.29 Dialogprogrammierung.....137
4.30 Eingabefunktion GET.....137
4.31 Eingabefunktion INPUT.....138
4.32 Warteschleifen.....141
4.33 Fehlerbehandlung.....143
4.34 Joystick und Paddleansteuerung.....146
4.35 Selektion von Peripheriegeräten.....147

4.36 Funktionen....148
4.36.1 Numerische Funktionen....148
4.36.2 ABS....148
4.36.3 RND....149
4.36.4 SGN....150
4.36.5 SIN....151
4.36.6 COS....152
4.36.7 TAN....152
4.36.8 ATN....153
4.36.9 EXP....153
4.36.10 INT....153
4.36.11 LOG....154
4.36.12 ASC....155
4.36.13 LEN....156
4.36.14 CHR$....157
4.36.15 LEFT$....157
4.36.16 RIGHT$....158
4.36.17 MID$....158
4.36.18 VAL....159
4.36.19 STR$....160
4.36.20 Definition von numerischen Funktionen....160
4.36.21 Systembezogene Funktionen....161
4.36.22 PEEK....161
4.36.23 POKE....162
4.36.24 HIMEM, LOMEM....163
4.36.25 CALL....164
4.36.26 USR....164

4.37 Der Programmtest....165
4.37.1 Die modulare Programmierung....165
4.37.2 Einfügen von PRINT-Anweisungen....166
4.37.3 TRACE, NOTRACE....167

**5 Die Programmierung und Anwendung der Maus....169**

5.1 Die Apple-Maus....171
5.2 Die Programmierung der Maus in BASIC....172
5.3 MousePaint....175
5.4 Werkzeuge in MousePaint....177
5.5 Architekturplanung auf dem Apple IIc....179

**6 Der Druckerbetrieb unter BASIC.....185**

**7 Grafik mit Applesoft-BASIC.....195**

7.1 Niedrigauflösende Grafik.....197
7.1.1 Zeichnen mit PLOT.....199
7.1.2 Bildpunkte ermitteln mit SCRN.....201
7.1.3 Horizontale Linien zeichnen mit HLIN.....202
7.1.4 Vertikale Linien mit VLIN.....203

7.2 Hochauflösende Grafik.....204
7.2.1 HiRes Grafikseite 1.....204
7.2.2 Farben setzen mit HCOLOR .....205
7.2.3 HPLOT.....206

7.3 Speichern von Grafikseiten.....209
7.4 Bilder-Animation.....208
7.5 3D-Darstellungen.....212

**8 Das Betriebssystem DOS 3.3.....215**

8.1 Die Diskette.....217
8.2 Der Schreibschutz.....219
8.3 Umgang mit Disketten.....219

8.4 DOS 3.3.....220
8.4.1 Initialisieren mit INIT.....220
8.4.2 RENAME.....222

8.5 DOS-Befehle in Programmen.....226
8.5.1 Sequentielle Textdateien.....226
8.5.2 Erstellen einer EXEC-Datei.....228
8.5.3 Textdateien mit wahlfreiem Zugriff.....229
8.5.4 Übertragen von Binärdaten.....230

**9 Das Betriebssystem ProDOS.....233**

9.1 Was ist ProDOS?.....235
9.2 Laden von ProDOS.....236

9.3 System-Dienstprogramme.....237
9.3.1 Kopieren von Dateien.....237
9.3.2 Löschen von Dateien.....238
9.3.3 Umbenennen von Dateien.....238
9.3.4 Schreibschutz setzen/aufheben.....239
9.3.5 Kopieren einer Diskette.....239
9.3.6 Formatieren einer Diskette.....240
9.3.7 Katalog einer Diskette.....240
9.3.8 Sonstige Funktionen.....241
9.3.9 PREFIX setzen.....241
9.3.10 Sub-Directory erstellen.....242
9.3.11 Konvertieren des Diskettenformats.....243
9.3.12 Überprüfen einer Diskette.....243
9.3.13 Serielle Anschlüsse konfigurieren.....243

9.4 Anzeigen des Inhaltsverzeichnisses in ProDOS.....244
9.5 Startup-Programm.....247
9.6 Laden und Abspeichern von Daten.....248
9.7 FP und INT in ProDOS.....250
9.8 Formatieren der Diskette.....250
9.9 Eingabeformat in ProDOS.....250
9.10 Erstellen eines Subdirectory.....250
9.11 Aktivieren der RAM-Disk.....253
9.12 PR# und IN# in ProDOS.....254
9.13 Filepuffer bei der Datenübertragung.....254

**10 Der System-Monitor.....257**

10.1 Die Bedeutung des Monitors.....260
10.2 Starten des System-Monitors.....262
10.3 Speicheradressen anzeigen.....262
10.4 Speicherinhalte verändern.....263
10.5 Verschieben eines Speicherbereichs.....265
10.6 Vergleichen von zwei Speicherbereichen.....266
10.7 Starten von Monitorprogrammen.....266
10.8 Rechnen im System-Monitor.....267

Anhang A: Reservierte Worte des Applesoft-Interpreters.....269

Anhang B: Fehlermeldungen .....270
- in Applesoft-BASIC.....270
- in Integer-BASIC.....272

Anhang C: Maschinenunterprogramme.....275

Anhang D: ASCII-Codes.....277
- der deutschen Tastenbelegung.....277
- der US-Tastenbelegung.....279

Anhang E: OP-Code der CPU 65C02.....281

Anhang F: Assemblerlisting des $F800-ROM.....291

Stichwortverzeichnis.....320

# 1

# Die Apple-Geschichte

## 1 Die Apple-Geschichte

Als 1978 Steve Jobs und Steve Wozniak den ersten Apple-Computer in einer Garage zusammenbauten, wußte wohl noch niemand, welchen Erfolg sie damit haben würden. Mit den einfachsten Mitteln wurde das erste Kind der Apple-Familie nachts in einem Labor bei Hewlett-Packard entwickelt und erprobt, bis der Computer mit Stolz der Öffentlichkeit vorgestellt werden konnte.

Der erste Apple, der Apple I, konnte etwa zweihundertmal verkauft werden und hatte noch kein Gehäuse und keine Tastatur. Auch die anfallenden Daten mußten von einem Kassettenrecorder geladen und wieder in den Computer eingespielt werden.

Mit dem Apple II gelang der Apple Computer Inc. der Sprung zu einem der umsatzstärksten Computerhersteller der Welt. Der Apple II war mit 16, 32 oder 48 K Speicherbereich ausgestattet und wurde in Ganzzahl-BASIC oder Assembler programmiert. Er hatte nun ein Gehäuse, eine professionelle Tastatur und konnte mit einem Diskettenlaufwerk ausgebaut werden.

Der Apple IIplus war eine Weiterentwicklung des Apple II. Neben Gleitkomma-BASIC und einer Autostart-Routine hatte man auch das Betriebssystem in einigen Punkten geändert und verbessert. Viele Fremdhersteller hatten schnell erkannt, daß der Apple IIplus ein erfolgreicher Computer wird. Darum wurde eine Vielzahl von Erweiterungen und Peripherie angeboten. Auch die Softwarehersteller entwickelten für den Apple II und IIplus eine Menge an Anwenderprogrammen und Spielen. Der Apple IIplus wurde wegen seiner ausgereiften Hard- und Software von vielen Firmen kopiert.

Für den deutschen und europäischen Markt wurde der Apple IIeuroplus entwickelt. Er verfügte über ein 220-Volt-Netzteil, und die Hardware war auf die Netzfrequenz von 50 Hertz abgestimmt. Die Erweiterungsmöglichkeiten der Apple-II-Serie hatte zur Folge, daß der Apple bald bei einer Vielzahl von Überwachungs-, Steuer- und Prüfaufgaben eingesetzt wurde.

Der Apple IIe kam 1983 auf den Markt. Viele der Mängel des Apple II, IIplus, IIeuroplus wurden beim Apple IIe berücksichtigt und konnten verbessert werden. Aus Gründen der Kompatibilität zu anderen Apple-Rechnern mußten einige Neuerungen als Erweiterungs-

karten entwickelt werden. Über diese steckbaren Karten war es so möglich, 80 Zeichen auf einer Bildschirmzeile darzustellen oder den Speicherbereich auf 128 KByte auszubauen.

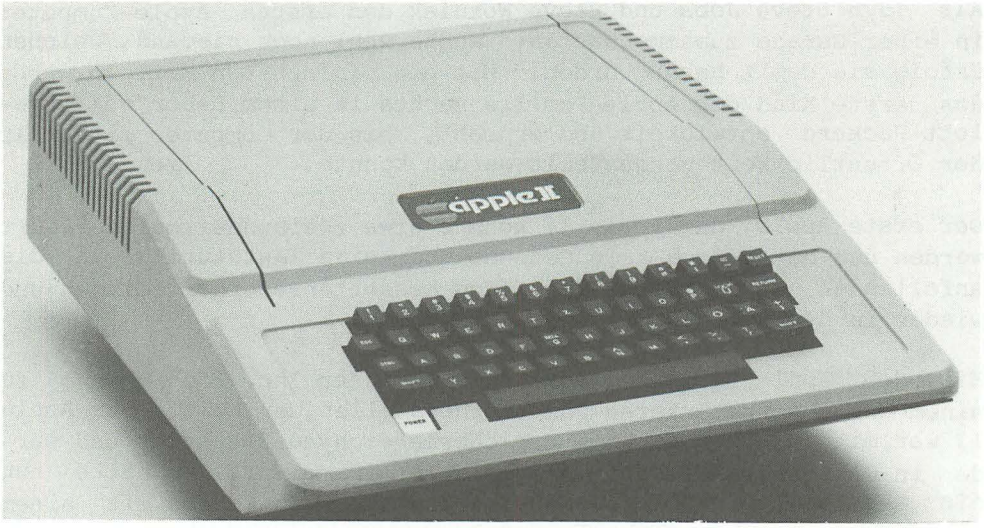



Abb. 1-1: Der Apple IIeuroplus

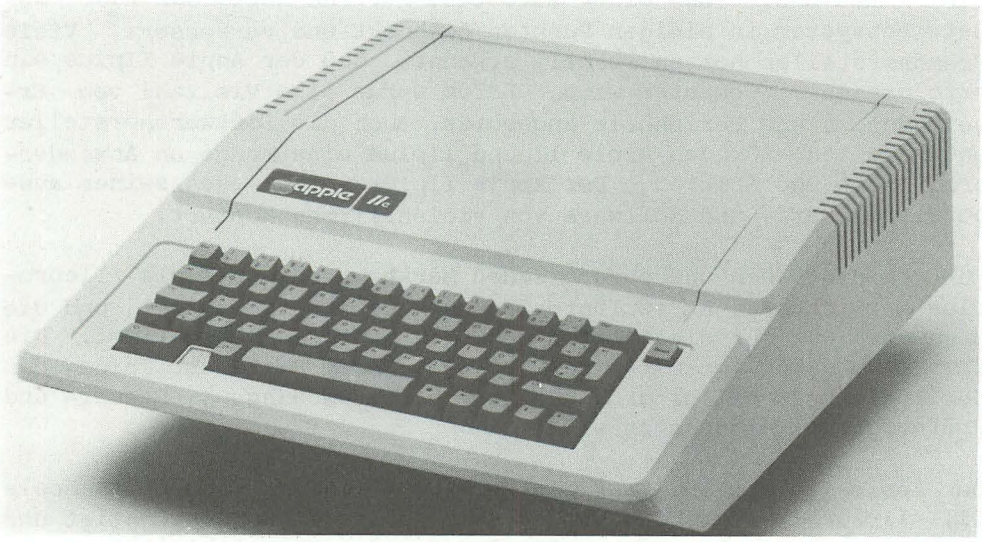



Abb. 1-2: Der Apple IIe

Um auch den professionellen Anwendungen gewachsen zu sein, wurde der Apple III gefertigt. Die Hardware wurde so konzipiert, daß auch große Datenmengen leicht verarbeitet werden können. So entwickelte man für diesen Rechner ein Festplattenlaufwerk mit 5 MegaByte Speicherkapazität. Auch die interne Rechnerstruktur war in der Verarbeitung schneller. Die 128 KByte Arbeitsspeicher ließen eine hohe Rechnerleistung zu, und im Computer war bereits ein Diskettenlaufwerk integriert. Die Softwareprodukte wurden ganz auf die Anwendungen im Büroalltag zugeschnitten.

Abb. 1-3: Apple III mit Festplatte und Monitor

Mit LISA von Apple wurde erstmalig eine völlig neue Technologie für den Computerlaien entwickelt. Der Computer sollte es einem Geschäftsmann oder Büroangestellten ermöglichen, auf einfachste Weise mit dem Computer zu arbeiten. Mit einem Softwarepaket, das Textverarbeitung, Tabellenkalkulation, Geschäftsgrafik, Listener-

stellung, Planung und Projektierung zuläßt, wird LISA zu einem effektiven Arbeitsmittel. Auf dem hochauflösenden Grafikbildschirm werden dem Bediener Arbeitswerkzeuge des Büroalltags angeboten. Arbeitsblätter können auf dem Schreibtisch ausgebreitet werden, Daten werden in Aktenordner abgelegt oder in einen Papierkorb geworfen. Es wird ausgeschnitten, kopiert und wieder eingesetzt. Der Anwender soll diesen Computer als elektronischen Arbeitsplatz benutzen, genauso wie vorher den Aktenschrank und die Schreibmaschine. Apple Computer Inc. ist mit diesem Computer dem Anwender ein großes Stück nähergekommen.

Abb. 1-4: LISA

Mit dem Macintosh von Apple begann ein neues Zeitalter der Computer. Neben der Steuerung mit der "Maus" konnte der "Mac" besonders mit seiner Bedienerfreundlichkeit aufwarten. Anhand bestimmter Symbole, die mit Hilfe der Maus anzuwählen sind, kann der

Computer auf einfache Weise gesteuert und bedient werden. Diese Technologie ermöglicht auch dem Laien in kürzester Zeit eine sinnvolle Nutzung des Computers.

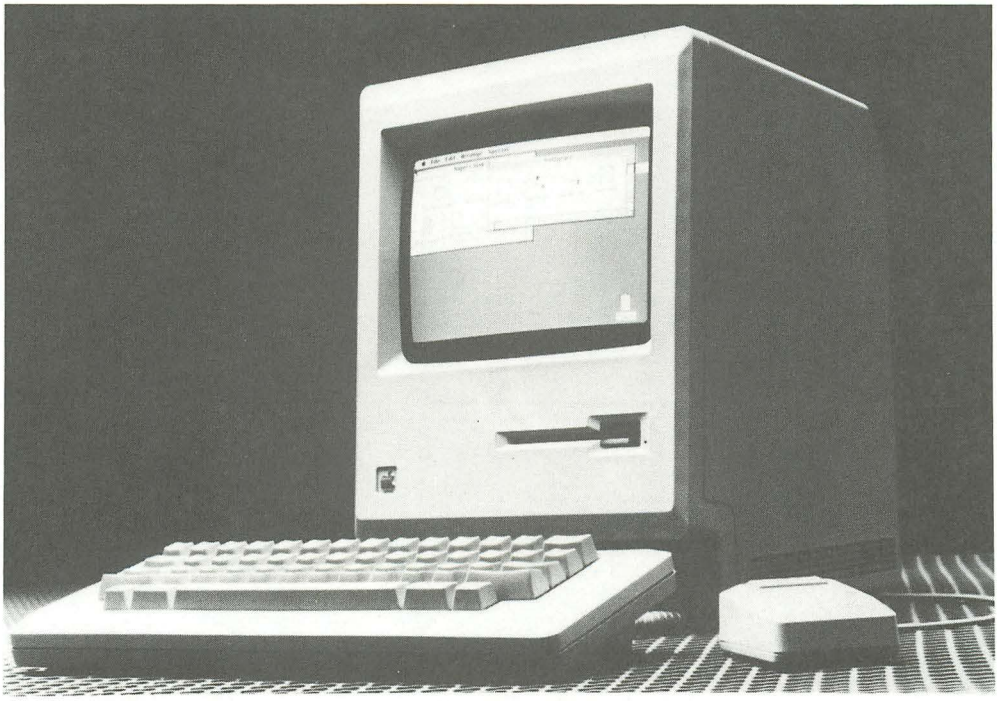


Abb. 1-5: Der Macintosh

Aufgrund vieler Marktstudien und Untersuchungen der Anwendergewohnheiten wurde der Apple IIc in seiner heutigen Form auf den Markt gebracht. All die Vorteile der Apple II-Serie sind in diesem Modell vereint worden. Die Erweiterungen, die bei den Vorgängermodellen erforderlich waren, wurden in dieses Gerät bereits eingebaut. Durch eine hohe Integration der Bauteile konnte der Computer portabel und kleiner gestaltet werden.

Um Ihren Apple IIc problemlos bedienen zu können, sollten Sie die folgenden Seiten genau durcharbeiten. So wird Ihnen das Buch wichtige Informationen für die tägliche Arbeit geben.

# 2

# Das Apple-IIc-System

## 2 Das Apple-IIc-System

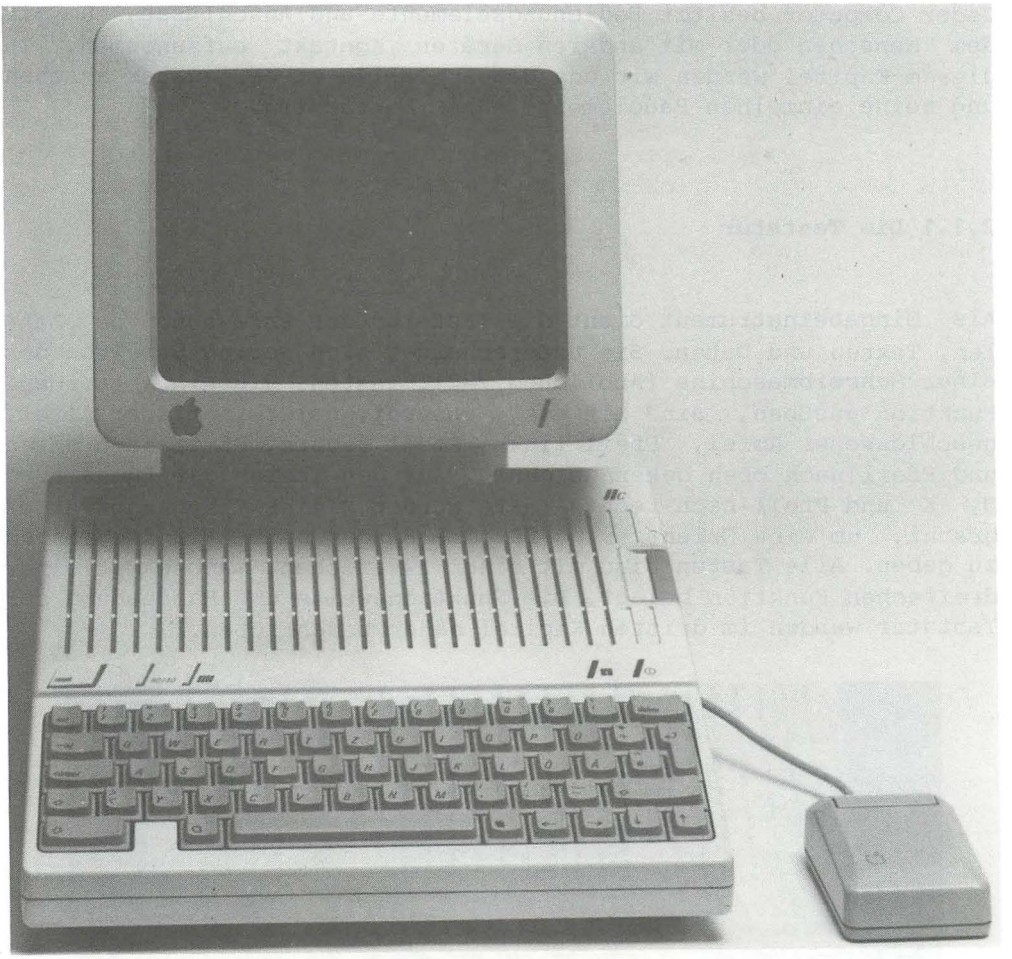


Abb. 2-1: Das Apple-IIc-System

Sicher haben Sie Ihren Apple IIc nicht lange in seiner Verpackung belassen. Doch bei aller Freude über Ihr neues Computersystem sollten Sie nicht vergessen, die einzelnen Teile zu überprüfen. Jeder Apple IIc wird mit einer beigelegten Packliste ausgeliefert. Nach dieser sollten Sie die Anzahl der Teile und deren Bezeichnung kontrollieren. Fehlende Teile sollten Sie schnellstens dem Händler oder Lieferanten Ihres Computers mitteilen.

## 2.1 Erster Kontakt mit dem Computer

Jeder Computer besitzt Bedienungselemente und Anschlüsse, um mit dem Menschen oder mit anderen Geräten Kontakt aufzunehmen. In diesem Kapitel werden wir uns den Apple IIc etwas genauer ansehen und seine einzelnen Bauelemente näher betrachten.

### 2.1.1 Die Tastatur

Als Eingabeinstrument dient die Tastatur zur Erfassung von Zahlen, Texten und Daten. Sie unterscheidet sich geringfügig von der einer Schreibmaschine (Abbildung 2-2). Tasten, die eine besondere Funktion ausüben, sind mit esc, control, delete, offener Apfel, geschlossener Apfel, Pfeil links, Pfeil rechts, Pfeil nach unten und Pfeil nach oben gekennzeichnet. Auf den Tasten der Buchstaben D, K und Pfeil-nach-rechts-Taste wurden kleine Erhebungen angebracht, um eine Orientierung für die Finger beim Blindschreiben zu geben. Alle Tasten sind mit einer zweifachen, manche mit einer dreifachen Funktion belegt. Die Funktionsweise und Handhabung der Tastatur werden im dritten Kapitel näher beschrieben.

Abb. 2-2: Die Tastatur

### 2.1.2 Schalter und Taster

Links über der Tastatur befindet sich ein Taster, der mit reset gekennzeichnet ist (Abbildung 2-2). Dieser, in Verbindung mit der CONTROL-Taste und der offenen Apfeltaste gedrückt, bewirkt einen System-Reset. Reset heißt nichts anderes als zurücksetzen. Der Computer wird in seinen Einschaltzustand gebracht.

Links neben der Reset-Taste sehen wir zwei Schalter, die zum einen mit 80/40 und zum anderen mit kleinen Quadraten beschriftet sind (Abbildung 2-2). Die linke Taste dient zum Umschalten von 40 auf 80 Zeichen je Bildschirmzeile. Die rechte Taste ist die Umschalttaste von US-Tastatur auf deutsche Tastenbelegung.

Auf der Rückseite des Computergehäuses, wie in Abbildung 2-2 dargestellt, befindet sich ein kleiner grüner Kippschalter. Dieser Netzschalter dient zum Ein- und Ausschalten des Computers. Mit diesem Schalter wird nur der Rechner ein- und ausgeschaltet, nicht aber der Netztransformator.

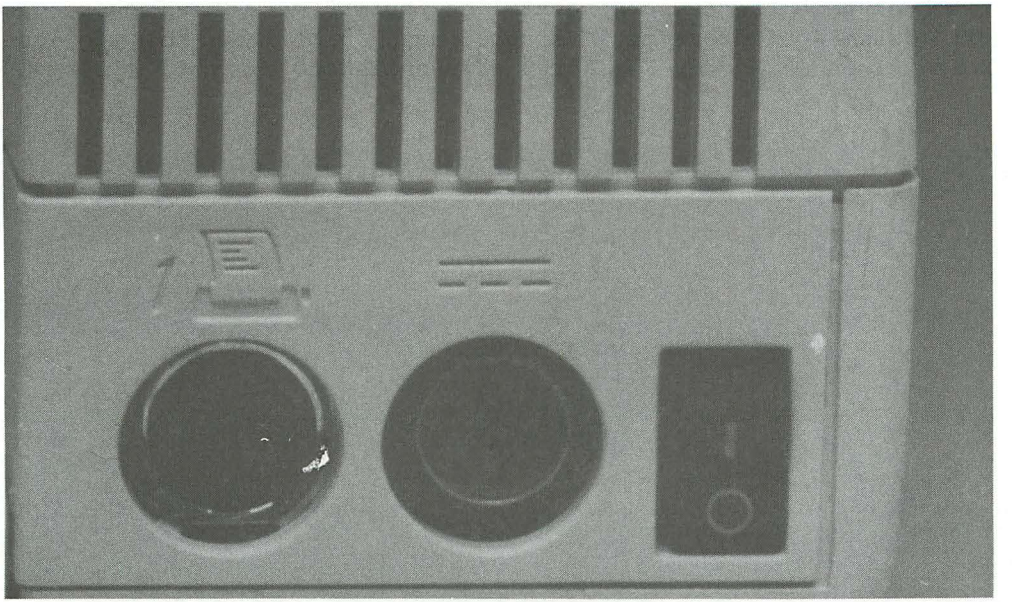


Abb. 2-3: Netzschalter und Netzbuchse

### 2.1.3 Die Kontrollampen

Die Kontrollanzeigen befinden sich rechts über der Tastatur. Die linke Anzeige, die symbolisch mit einer Diskette gekennzeichnet ist, zeigt an, wann das interne Diskettenlaufwerk Daten liest oder auf die Diskette schreibt. Sie leuchtet bei Betrieb des Diskettenlaufwerks rot auf. Die rechte Anzeige soll die Betriebsbereitschaft des Computers signalisieren. Nach dem Einschalten des Computers leuchtet die Lampe grün.

### 2.1.4. Lautstärkeregler, Kopfhöreranschluß

Wenn Sie auf die rechte Seite Ihres Apple IIc schauen, entdecken Sie zwei weitere Bedienungselemente: den Lautstärkeregler und den Kopfhöreranschluß. Der Lautstärkeregler, der nur zum Teil zu sehen ist, ermöglicht Ihnen, die Lautstärke des eingebauten Lautsprechers nach Ihren Wünschen einzustellen. Die senkrechten Linien, die auf dem Drehregler angebracht sind, zeigen Ihnen optisch je nach Länge die eingestellte Lautstärke. Die Buchse, die mit einem Kopfhörersymbol versehen ist, gibt Ihnen die Möglichkeit, einen Ohr- oder Kopfhörer anzuschließen.

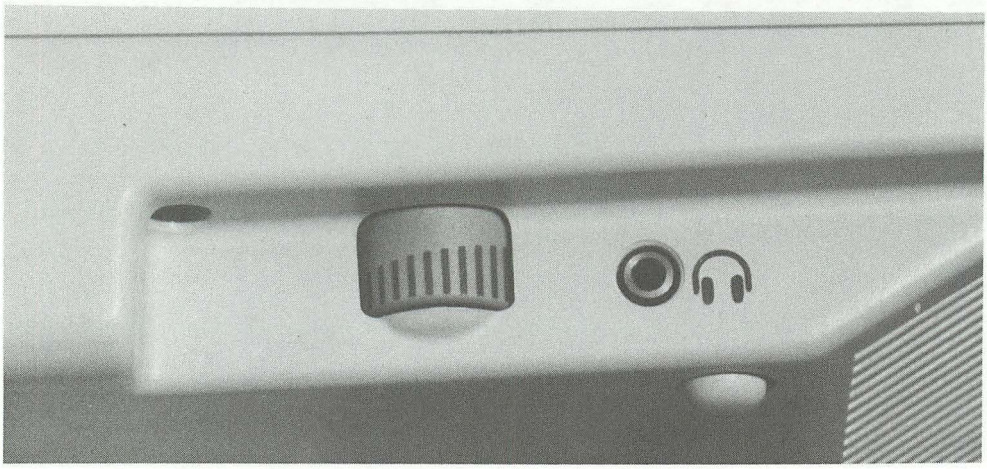


Abb. 2-4: Lautstärkeregler und Kopfhöreranschluß

### 2.1.5 Das eingebaute Diskettenlaufwerk

Auf der rechten Seite Ihres Computers befindet sich der Einschub für Disketten (Abbildung 2-4), das eingebaute Diskettenlaufwerk. Drücken Sie auf die Einschubklappe und ziehen Sie sie nach oben. Das Laufwerk ist jetzt geöffnet und Sie können Ihre Diskette einlegen und das Diskettenlaufwerk wieder schließen.

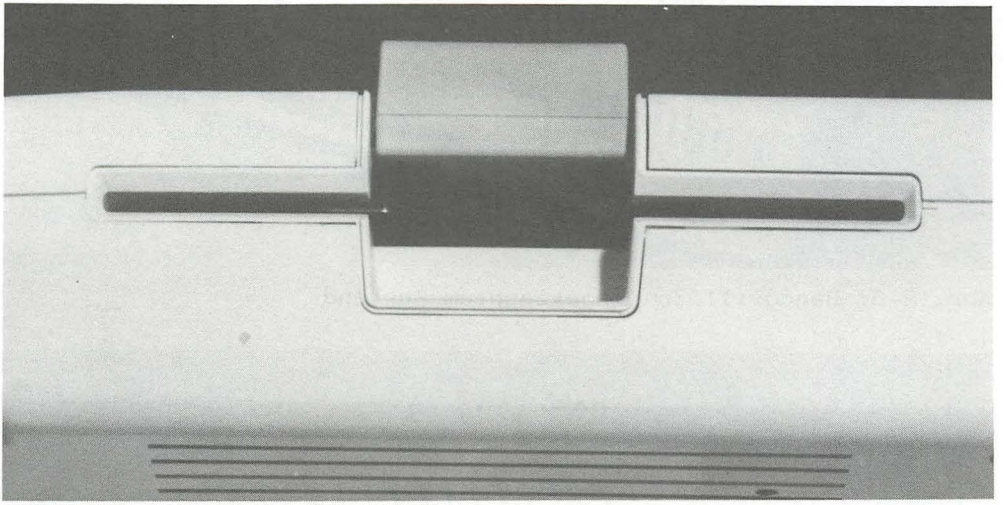


Abb. 2-5: Das eingebaute Diskettenlaufwerk

### 2.1.6 Der Handgriff

Wenn Sie das Gerät umdrehen, können Sie eine Menge von Anschlüssen mit darüberliegenden Symbolen sehen. In einer U-förmigen Vertiefung liegt eingeklappt (Abbildung 2-6), ein Handgriff, mit dem Sie Ihren Apple IIc wie einen kleinen Aktenkoffer tragen können. Um eine gute Durchlüftung des Computers zu gewährleisten, muß der Tragegriff bei Betrieb immer ausgeklappt sein (vgl. Abbildung 2-7). Außerdem können Sie durch die Schrägstellung die Tastatur leichter bedienen.

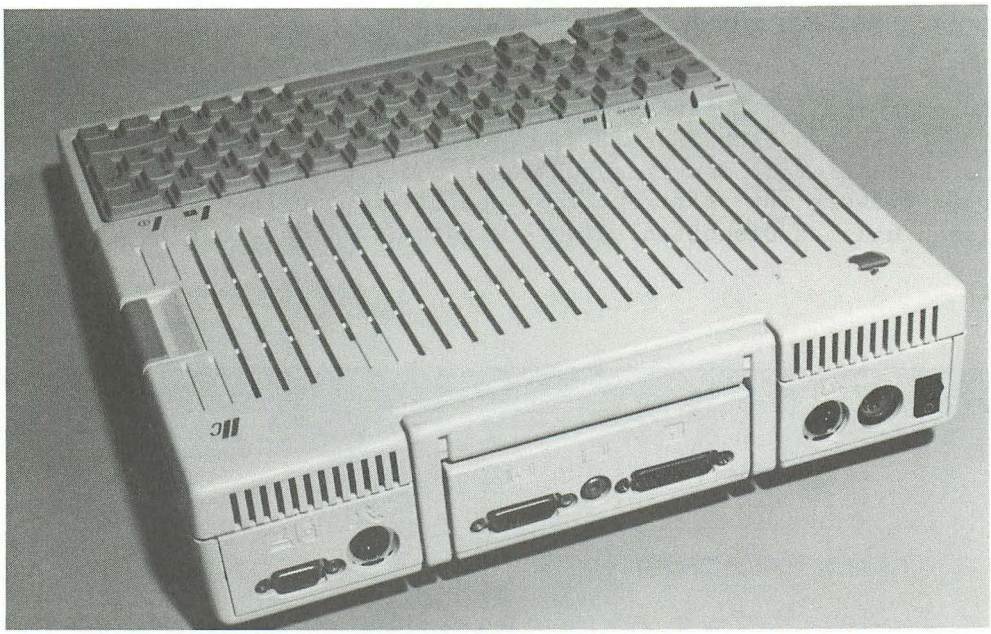


Abb. 2-6: Handgriff in eingeklapptem Zustand

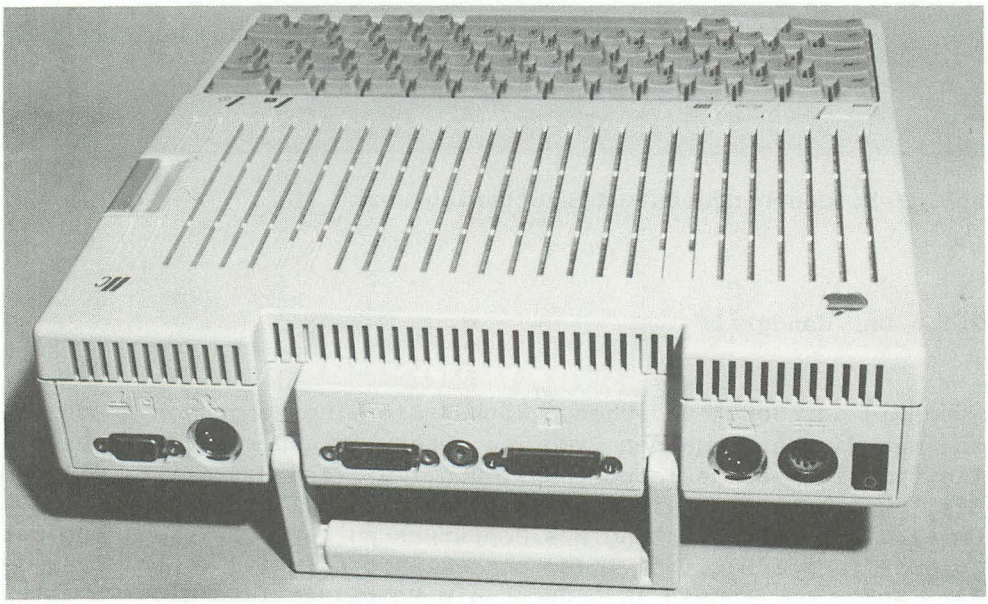


Abb. 2-7: Handgriff ausgeklappt

### 2.1.7 Joystick- und Maus-Anschluß

Von links nach rechts sehen Sie als erstes einen länglichen Anschluß mit neun Kontaktlöchern (Abbildung 2-8). Er dient zur Aufnahme des Steckkontaktes für Joystick, Drehregler und die Maus. Der Joystick wird auch X/Y-Steuerhebel genannt, weil mit ihm eine vertikale und horizontale Steuerung auf dem Bildschirm möglich ist. Dasselbe gilt auch für die Maus. Mit den Drehreglern dagegen ist eine Steuerung nur auf einer Ebene möglich.

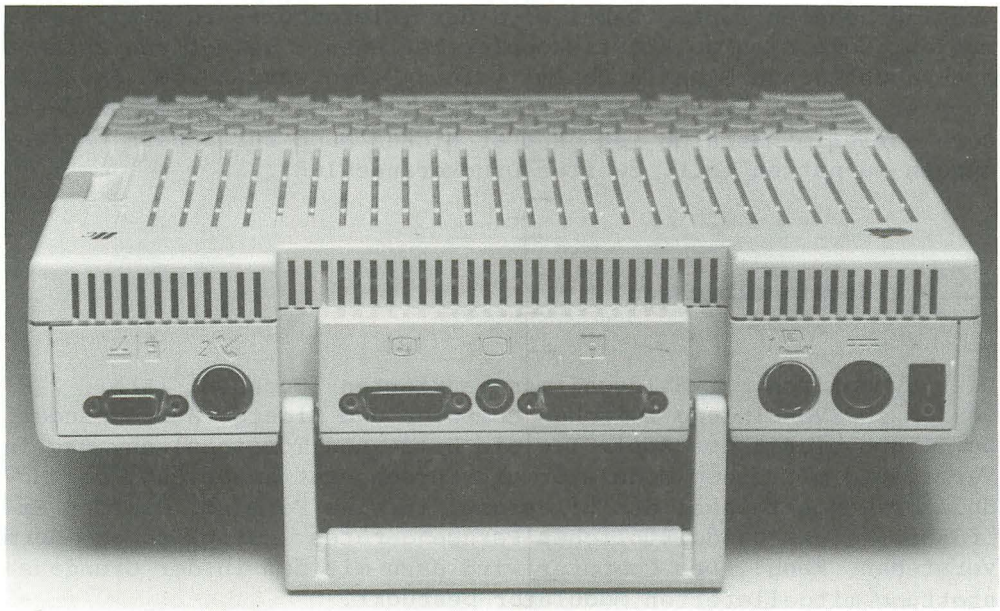


Abb. 2-8: Die Rückseite des Apple IIc

### 2.1.8 Serieller Anschluß 2

Die mit einem Telefonhörer und einer "2" beschriftete Buchse (Abbildung 2-8) dient zum Anschluß eines Modems oder Akustikkopplers. Modem ist die Abkürzung Mo-dulator und Dem-odulator. Es dient zur Übertragung von Daten über das Telefonnetz. Diese werden aufgeschlüsselt (moduliert), über die Telefonleitung an den Empfänger geleitet und dort wieder entschlüsselt (demoduliert).

Der Unterschied zwischen Modem und Akustikkoppler besteht in der Verbindung des Computers mit dem Telefonnetz. Über das Modem ist der Rechner direkt mit dem Telefonnetz verbunden und darf daher nur von einem Fernmeldehandwerker installiert werden.

Anders dagegen der Akustikkoppler, wo sich dieses Übertragungsproblem beheben läßt. Dabei wird der Telefonhörer in zwei Gummimuffen, die sich am Akustikkoppler befinden, gelegt und kann so die akustischen Signale übermitteln. Diese werden vom Akustikkoppler in für den Computer verständliche Signale umgewandelt. Der serielle Anschluß ist darüber hinaus auch für andere Anwendungen zu nutzen (z.B. zweiter Druckeranschluß).

### 2.1.9 Videosignalbuchsen

Die symbolisch mit Bildschirmen markierten Buchsen (Abb. 2-8) werden zur Verbindung zwischen Computer und Sichtgerät verwendet. Zum einen bietet der Apple IIc ein Bild- und Tonsignal an, das in Verbindung mit einem Modulator ein Fersehgerät anspricht. Der Modulator wandelt dabei das Bildsignal in eine Form um, die der Empfänger eines handelsüblichen Fernsehgerätes europäischer Norm verstehen kann. Der Computer wird dazu mit dem in der Grundausstattung mitgelieferten Modulator bestückt.

Zum anderen liefert die Koaxialbuchse, die Sie in der Mitte sehen, das Videosignal für einen Monitor, der über ein Koaxialkabel angeschlossen werden kann.

### 2.1.10 Diskettenlaufwerk-Anschluß

Rechts neben der Videosignalbuchse befindet sich der Anschluß für das externe Diskettenlaufwerk (Abbildung 2-8). Dieses ist bei umfangreichen Programmiersprachen und Programmen oft unerläßlich.

### 2.1.11 Serieller Anschluß 1

Dieser Anschluß dient in Verbindung mit einem mehradrigen Kabel der Ansteuerung eines Druckers oder Plotters (Abbildung 2-8). Über diese Leitungen sendet der Computer alle Daten, die für die Übermittlung von Druckzeichen erforderlich sind.

Dieser serielle Anschluß kann auch für andere Aufgaben benutzt werden, wie zum Beispiel die Steuerung eines Kleinroboters.

### 2.1.12 Netzanschluß

Die Anschlußbuchse neben dem Netzschalter verbindet den Transformator mit dem Computer (Abbildung 2-3). Die von 220 Volt Wechselspannung auf 15 Volt transformierte und gleichgerichtete Spannung wird über ein steckbares Kabel an den Computer geleitet. Das integrierte Schaltnetzteil wandelt die ankommende Gleichspannung in die erforderlichen Teilspannungen um.

### 2.1.13 Netzteil/Transformator

Die Verbindung zwischen Computer und der Steckdose stellt das Netzteil her. Es dient zur Transformation der Spannung von 220 Volt Wechselspannung auf 15 Volt Gleichspannung. Das Kabel zur Steckdose wird am Netztrafo eingesteckt. Das Kabel zum Computer ist mit dem Netztrafo fest verbunden und wird in die vorgesehene Buchse eingesteckt.

### 2.1.14 Die Lüftungsschlitze

Das Gehäuse Ihres Apple IIc ist mit einer Vielzahl von Lüftungsschlitzen versehen. Der Grund liegt in der Wärmeentwicklung der eingebauten Elektronikbauteile. Diese Lüftungseinrichtung befindet sich sowohl an der Ober- als auch auf der Unterseite. Der Computer darf deshalb nicht auf weichen Untergrund gestellt oder abgedeckt werden, da sonst die Luftzirkulation nicht mehr gewährleistet wäre.

### 2.1.15 Die Unterseite

Auf der Unterseite finden Sie ein Label, auf dem neben dem Herstellungsland und den Patenten, die für dieses Gerät angemeldet sind, noch eine mehrstellige Zahl ausgewiesen ist. Diese Geräte- oder Seriennummer sollen Sie auf jeden Fall notieren. Da jeder Apple IIc eine eigene Nummer erhalten hat, kann sie bei einem eventuellen Diebstahl von Nutzen sein.

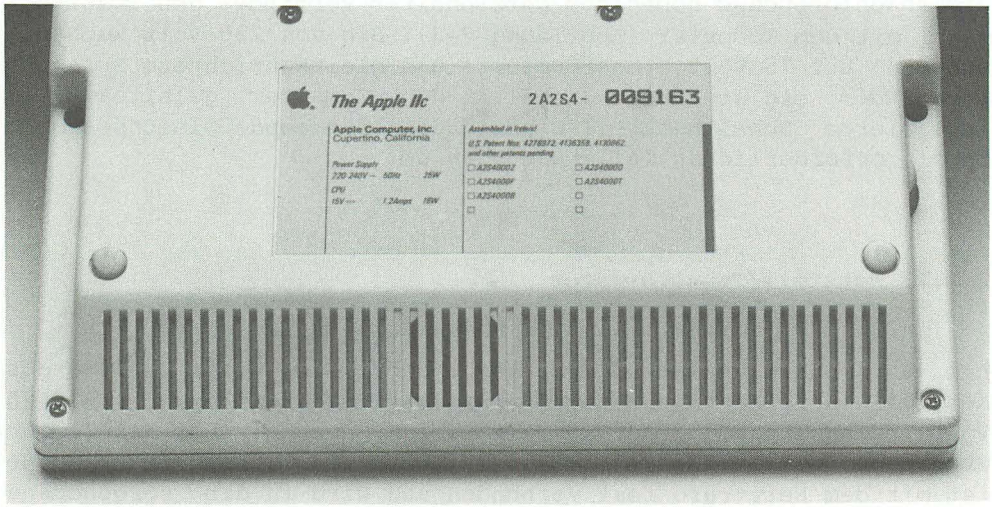



Abb. 2-9: Das Typenschild

### 2.1.16 Der PAL-Modulator/Adapter

Um ein handelsübliches Fernsehgerät anschließen zu können, muß der mitgelieferte PAL-Modulator/Adapter installiert werden (Abbildung 2-10). Dies wird durch das Anstecken des Modulators an die Buchse mit dem abgebildeten Monitor und den drei Kreisen realisiert. Anschließend wird der Fernseher mit dem beigefügten Kabel an den Modulator/Adapter angeschlossen.

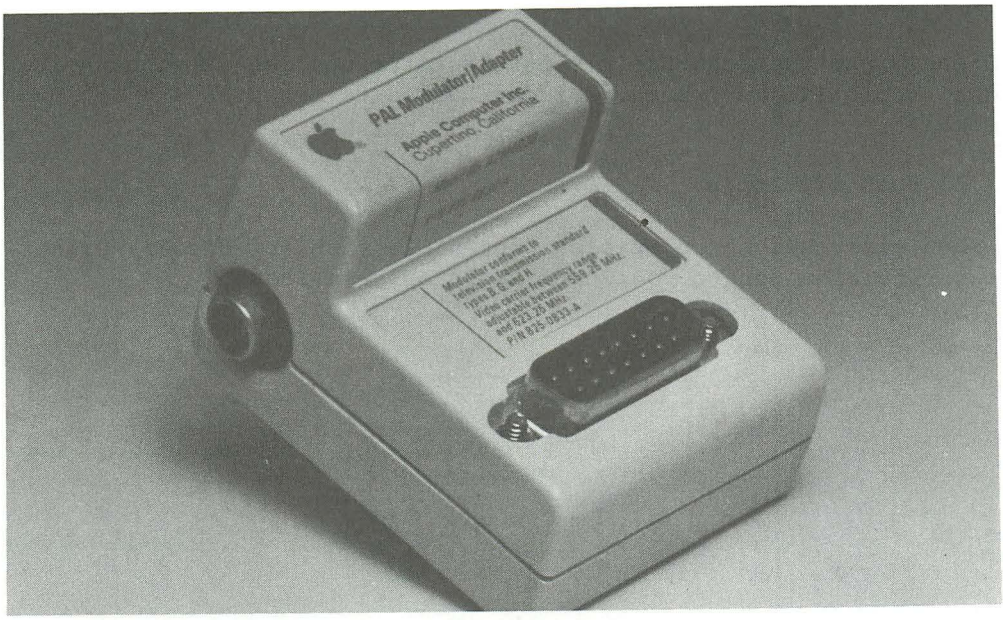



Abb. 2-10: Der PAL-Modulator/Adapter

### 2.1.17 Die Handbücher

Die beigepackten Handbücher sind eine hilfreiche Anleitung für den Anfänger und auch den, der bereits fertige Programme besitzt. Wer dagegen Applesoft-BASIC oder Maschinensprache programmieren möchte, sollte zuerst die entsprechenden Bücher erwerben, um die Leistung seines Apple IIc voll ausnutzen zu können.

### 2.1.18 Die System- und Einführungsdisketten

Der Apple IIc wird mit fünf Disketten ausgeliefert die das Arbeiten mit ihm wesentlich erleichtern. Die folgende Aufstellung stellt die Disketten kurz vor:

**1 a) Eine Einführung**

Diese Diskette macht Sie mit der Tastatur und den Besonderheiten des Apple IIc vertraut. Auf der Rückseite der Diskette befindet sich:

**b) Spaß mit Apple**

Neben einer Reihe von Spielen können Sie sich auf dieser Diskettenseite eine Sonate von Ihrem Apple spielen lassen.

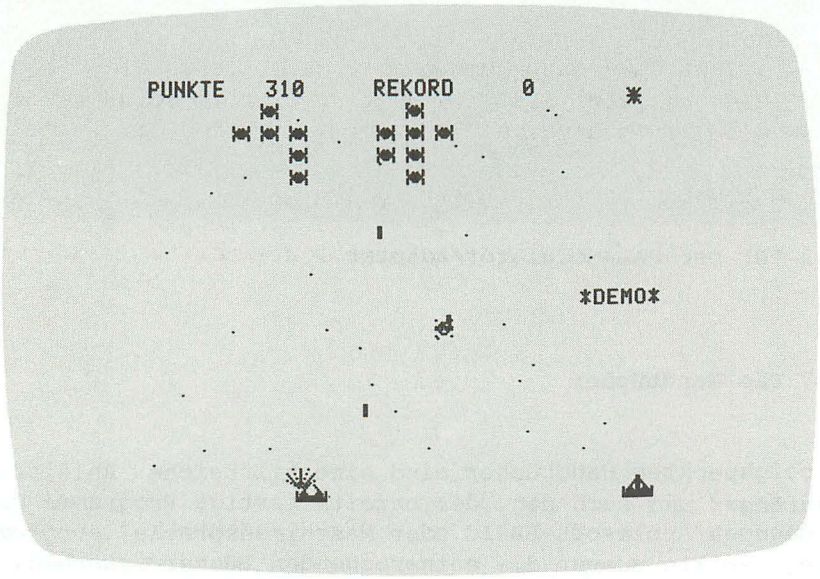



Abb. 2-11: Spaß mit Apple

## 2) Einführung in Apple LOGO

Die Diskette führt Sie in die Welt der Programmierung in LOGO. Anhand einfacher Beispiele machen Sie die ersten Schritte mit LOGO.

Auf der Rückseite dieser Diskette befinden sich Programme, die Ihnen einen kleinen Computerkrimi zeigen. Dabei lernen Sie etwas über Betriebssysteme und das Sichern von Daten.

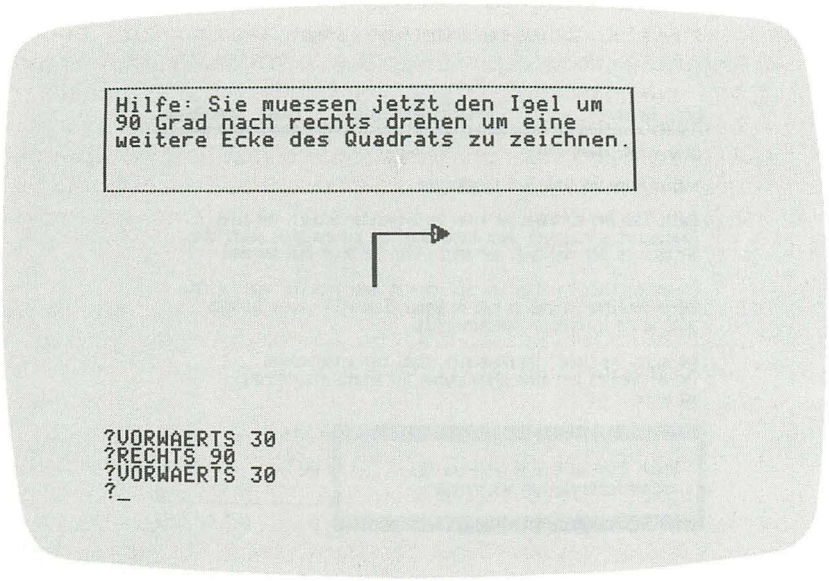



Abb. 2-12: Einführung in LOGO

## 3) Einführung in BASIC

In Form von einfachen Beispielen werden Ihnen auf dieser Diskette die ersten Grundlagen in BASIC gezeigt. Die Information reicht aber bei weitem nicht aus, um BASIC zu lernen. Die Diskette soll Ihnen nur einen Einblick in die Programmiersprache vermitteln.

## 4) Apple-Works-Demo

Um sich eine Vorstellung von den Möglichkeiten Ihres Apples zu machen, sollten Sie diese Diskette durcharbeiten. Sie zeigt Ihnen Textverarbeitung, Tabellenkalkulation und Datenverwaltung am Beispiel eines Fußballclubs. Die Diskette kann beidseitig verwendet werden. Im Modus 80 Spalten für die Darstellung auf einem Monitor und für 40 Spalten zur Darstellung auf einem Fernsehgerät.

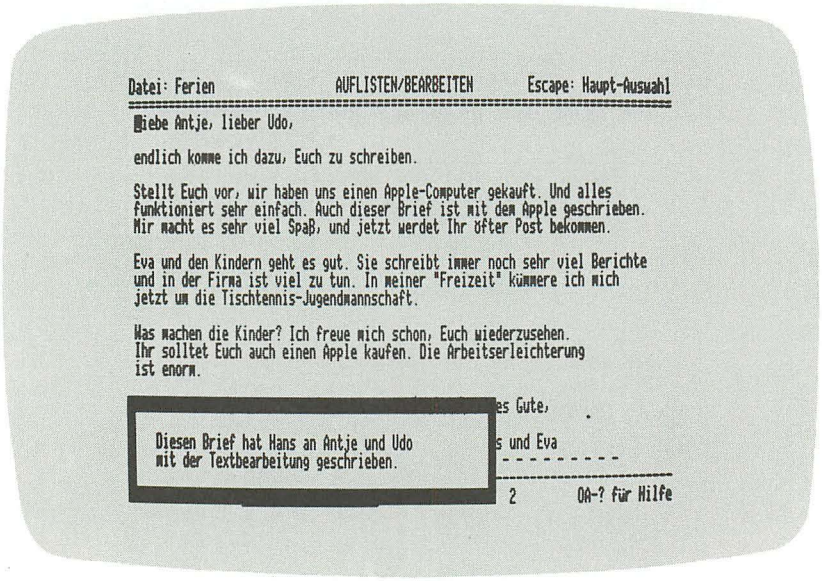



Abb. 2-13: Apple-Works-Demo: Textverarbeitung

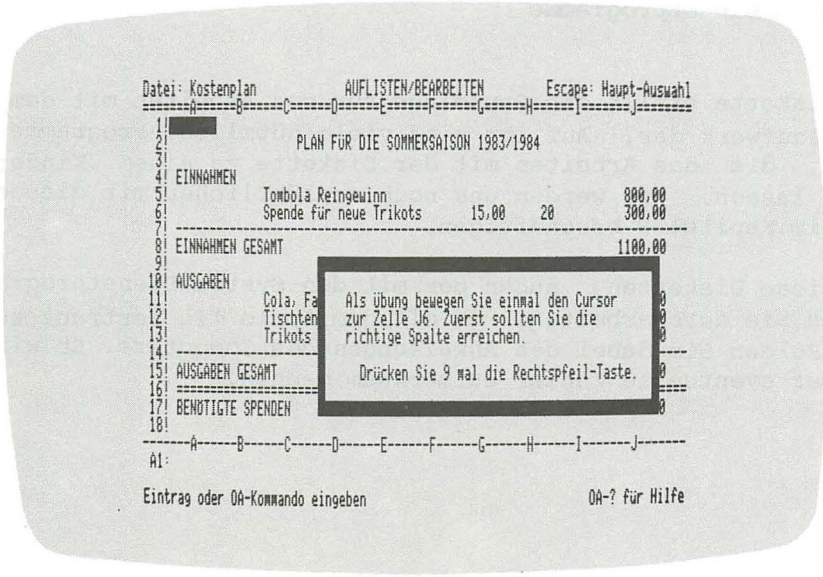



Abb. 2-14: Apple-Works-Demo: Tabellenkalkulation

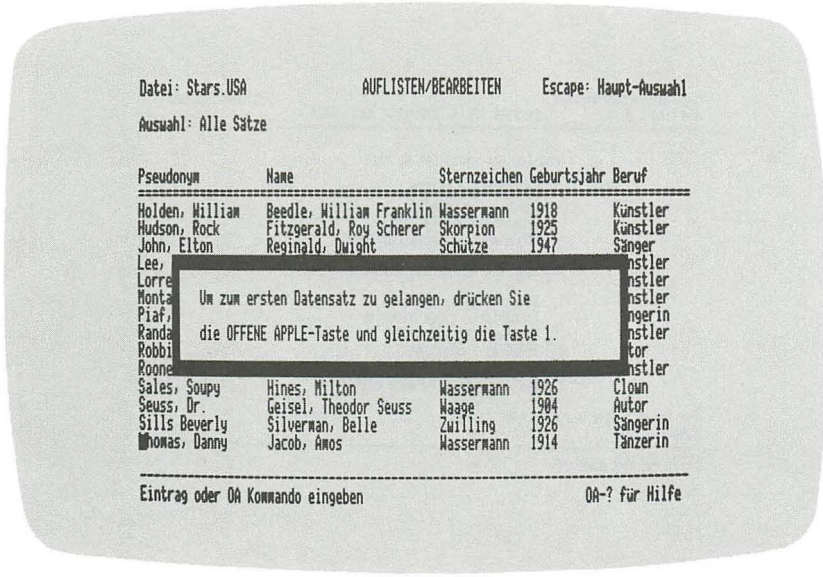



Abb. 2-15: Apple-Works-Demo: Datenverwaltung

## 5) System-Dienstprogramme

Die Diskette stellt die Grundlage für das Arbeiten mit dem Diskettenlaufwerk dar. Auf ihr sind viele nützliche Programme enthalten, die das Arbeiten mit der Diskette zu einem Kinderspiel werden lassen. Wir werden uns noch ausführlicher mit dieser Diskette im Kapitel 9 beschäftigen.

Alle diese Disketten, außer der mit den System-Dienstprogrammen, sollten Sie durcharbeiten, um mit dem Apple IIc vertraut zu werden. Folgen Sie dabei den Anweisungen des Computers. Er wird Sie auch auf eventuelle Fehler aufmerksam machen.

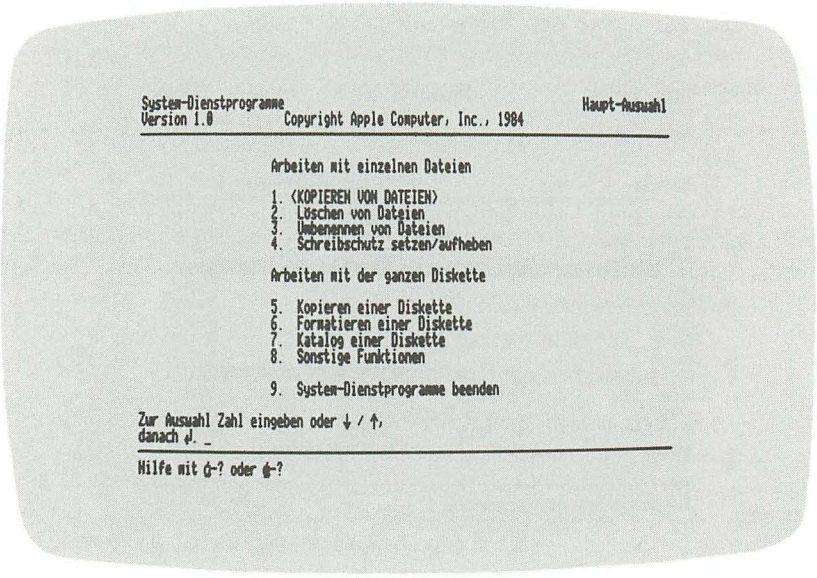



Abb. 2-16: Menü System-Dienstprogramme

## 2.2 Das Innenleben des Apple IIc

Wir haben uns den Apple IIc von außen genau angeschaut und wissen jetzt, was welcher Funktion dient. Was wir nicht sehen können, ist sein Innenleben. Um Ihre Neugier etwas zu befriedigen, habe ich den Apple IIc für Sie geöffnet. Sie selber sollten aber auf keinen Fall den Apple IIc öffnen. Der Garantieanspruch ginge dadurch nämlich verloren.

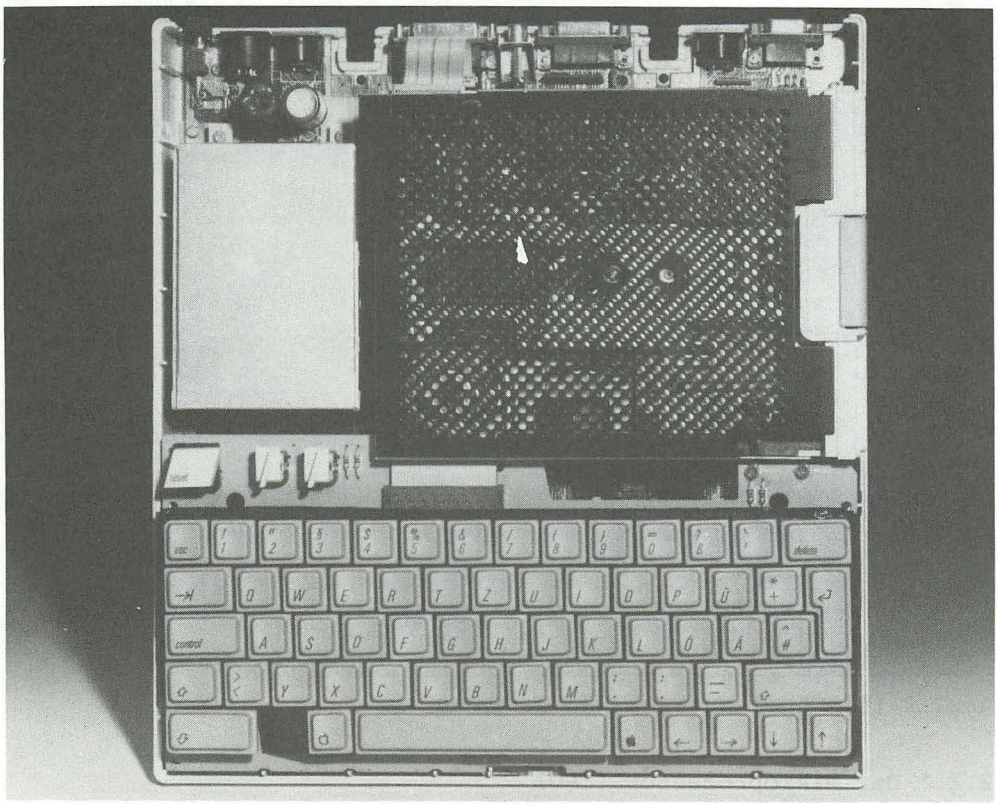


Abb. 2-17: Der geöffnete Apple IIc

Im Abbildung 2-17 sehen wir den geöffneten Apple IIc mit dem Diskettenlaufwerk und der Tastatur. Um auf die Systemplatine mit ihren Bauteilen blicken zu können, werden Diskettenlaufwerk und Tastatur ausgebaut. Abbildung 2-18 zeigt den Apple IIc ohne Laufwerk und Tastatur.

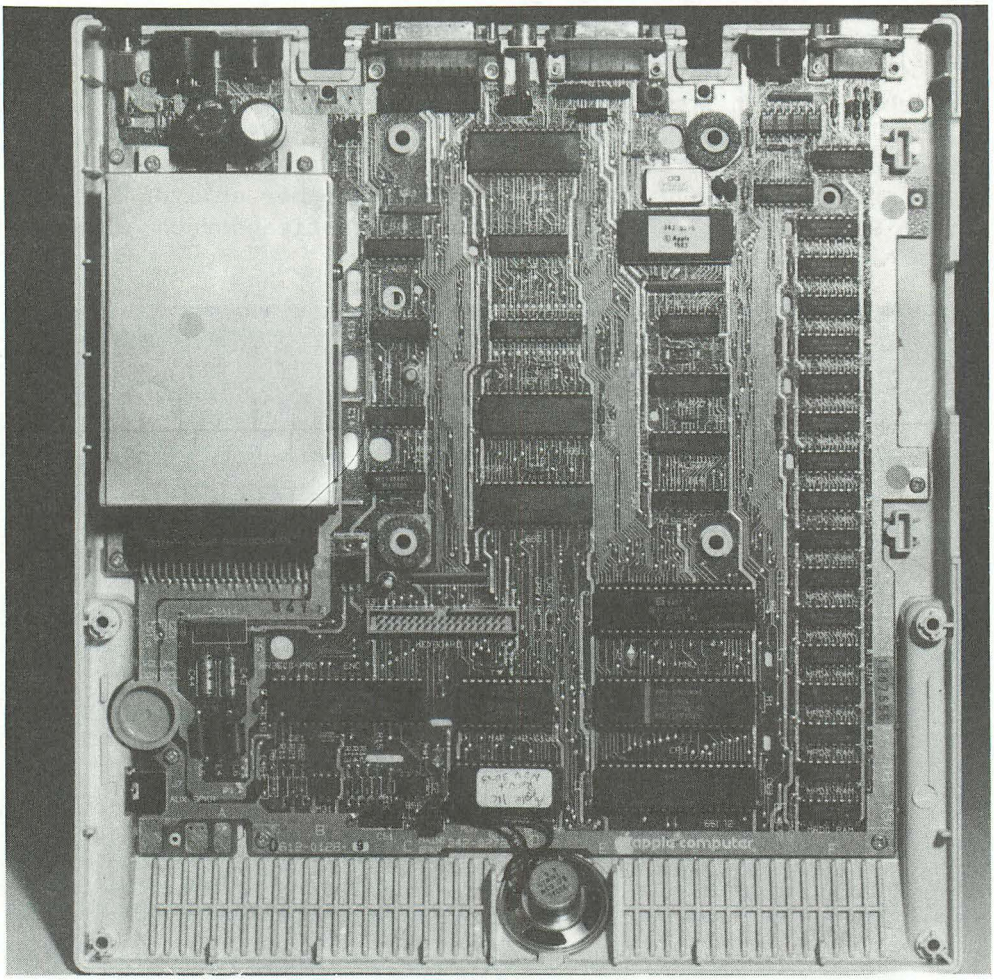


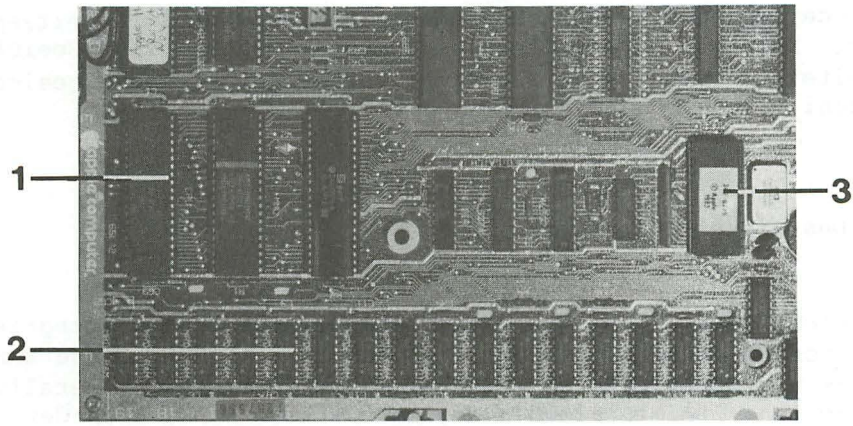
Abb. 2-18: Ohne Laufwerk und Tastatur

### 2.2.1 Der Mikroprozessor (CPU)

Wir haben auf den letzten Seiten viel über den Computer gelesen. Doch das eigentliche Herz des Computers sitzt ganz unscheinbar in einem schwarzen Kunststoffgehäuse auf der Systemplatine. Der Mikroprozessor oder die CPU (Central Processor Unit = Zentrale Rechnereinheit) erledigt die gesamte Verwaltung und Bearbeitung des Computers. Abbildung 2-19 zeigt uns den Mikroprozessor auf der Systemplatine.

Im Apple IIc wurde ein Mikroprozessor der 65er-Familie eingesetzt. Die genaue Bezeichnung heißt 65C02. Es handelt sich dabei um einen 8-Bit-Mikroprozessor mit einem 16-Bit-Adreßbus. Der Taktzyklus beträt ca. 1 Megahertz. Das bedeutet, daß das Herz des Mikroprozessors mit einer Taktfrequenz von 1 Million in der Sekunde schlägt. Der Leistungsverbrauch dieses Bausteins ist um ein Vielfaches herabgesetzt worden. Es handelt sich um einen CMOS-Typen (Complementary Metal Oxide Semiconductor), der wesentlich weniger Strom verbraucht als der 6502 in den anderen Apple-II-Modellen. Durch den geringeren Energieverbrauch wurde auch die Wärmeentwicklung vermindert.

Nicht nur hardwaremäßig unterscheidet sich der Mikroprozessor vom alten 6502-Modell. Es wurden auch einige Befehlsstrukturen hinzugefügt. Genauer können Sie das im Kapitel 10 bei der Maschinenprogrammierung nachlesen.




1 CPU, CMOS Mikroprozessor 65C02
2 64000-Bit-RAM-Baustein
3 16-KByte-ROM-Baustein

Abb. 2-19: CPU, RAM, ROM

### 2.2.2 Die Speicherbausteine

Um aber richtig arbeiten zu können, braucht der Mikroprozessor Speicherelemente. Man unterscheidet mehrere Arten von Speicherbausteinen:

- Festwertspeicher oder ROM (read only memory).

  Die in diesen Bausteinen enthaltenen Informationen stehen dem Computer sofort beim Einschalten zur Verfügung. Dort gespeicherte Programme können nicht überschrieben oder gelöscht werden. Der Computer benötigt diese Programme, um überhaupt starten zu können. In diesen ROMs sind das Monitorprogramm und der BASIC-Interpreter enthalten. Auch nach dem Ausschalten des Computers bleiben diese Daten erhalten.

- Flüchtiger Speicher oder RAM (random access memory).

  Diese Speicherbausteine dienen der Zwischenspeicherung von Daten, die der Mikroprozessor benutzt. Dieser Speicherbereich wird auch Arbeitsspeicher genannt. Alle Programme, die eingegeben oder geladen werden, befinden sich zunächst im Arbeitsspeicher. Nach dem Ausschalten des Computers gehen diese Speicherinhalte verloren, wenn sie nicht vorher auf Diskette gesichert wurden.

### 2.2.3 Das Diskettenlaufwerk

In unserem Computer ist das Diskettenlaufwerk bereits integriert. Um es vor elektromagnetischen Strahlen zu schützen (diese würden zu Lese- und Schreibfehlern führen), ist es mit einem Metallgitter umgeben. Beim Schließen des Laufwerks, senkt sich der Andruckfilz auf die Diskettenoberfläche. Er bildet das Gegenstück zum Schreib-/Lesekopf. Der Schreib-/Lesekopf des Laufwerks hat nach dem Einlegen der Diskette immer Kontakt mit deren Oberfläche. Aus diesem Grund sollten Sie die Diskette, wenn sie nicht benötigt wird, aus dem Laufwerk nehmen. (Abbildung 2-20 zeigt das Zusammenwirken der Laufwerkeinrichtungen.)

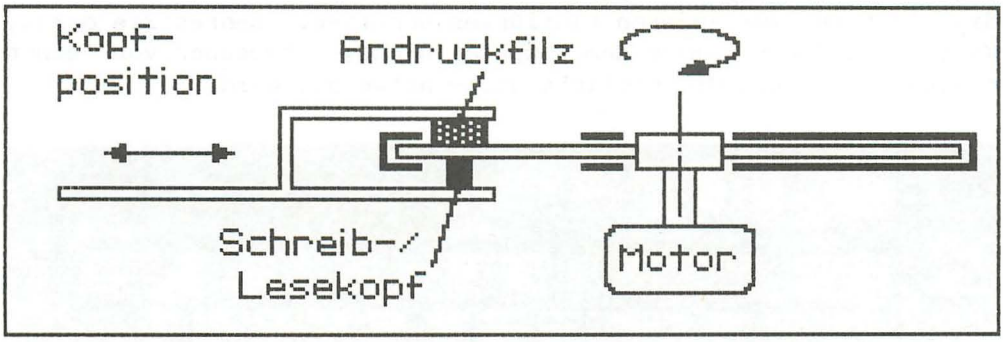



Abb. 2-20: Schreib-/Lesekopf und Andruckfilz

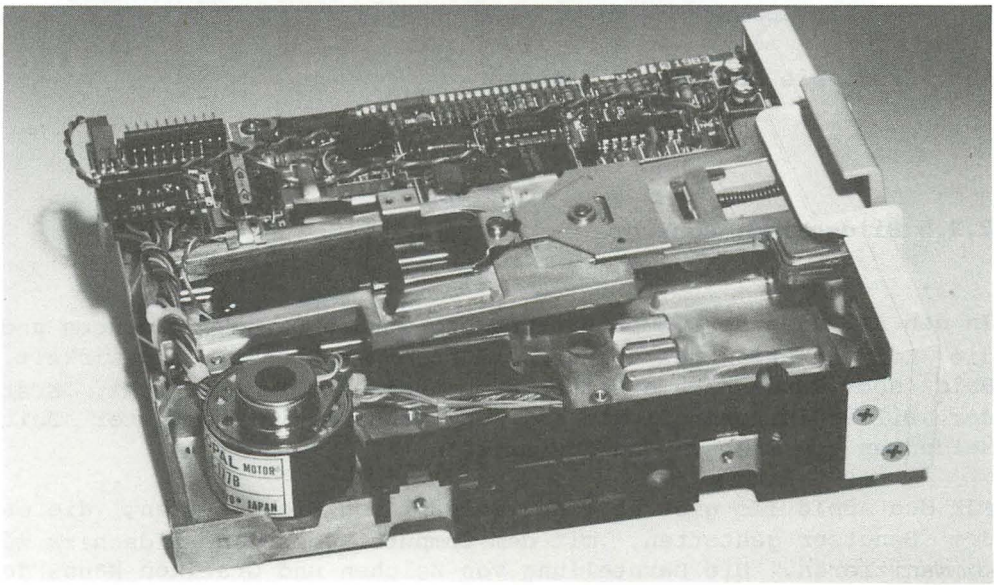


Abb. 2-21: Mechanik Diskettenlaufwerk

### 2.2.4 Die Tastatur

Bei abgenommener Tastaturabdeckung erkennen wir, daß zwischen Tasten und Trägerplatte eine Schutzfolie angebracht ist. Sie soll die Tastatur vor äußeren Einflüssen schützen. Denken Sie daran, Kaffee und Tee bekommen dem Computer nicht, abgesehen von einer teuren Reparatur, die möglicherweise notwendig wird.

Abb. 2-22: Die ausgebaute Tastatur

### 2.2.5 Bildschirm, Monitor

In den Anfängen der Computeranwendung wurden Daten, Meldungen und die Kommunikation des Computers über einen Drucker abgewickelt. Bald aber wurde nach einem anderen Ausgabemedium gesucht. Erst der Bildschirm ermöglichte es, dem Menschen in kürzester Zeit Meldungen vom Computer zu übermitteln.

Für den Apple IIc gibt es eine Vielzahl von Sichtgeräten, die es dem Benutzer gestatten, mit dem Computer über den Bildschirm zu kommunizieren. Die Darstellung von Zeichen und Grafiken kann, je nach Gerät, von unterschiedlicher Qualität sein.

Wir unterscheiden drei Arten von Sichtgeräten:

- Monitor oder Datensichtgerät
- Fernsehgerät (Schwarzweiß oder Farbe)
- LCD-Bildschirm oder Flachbildschirm

### 2.2.6 Der Monitor

Monitore werden unterteilt in Monochrom- (einfarbige) und Farbgeräte. Die Auflösung der Bildanzeige ist bei diesen Geräten höher als bei Fernsehempfängern. Das Bild wird schneller aufgebaut, was damit die Qualität beeinflußt. Man spricht auch von einer höheren Bandbreite. Bandbreiten zwischen 18 und 22 MegaHertz sind als gut bis sehr gut einzustufen. Beim Kauf eines Monitors sollten Sie auch auf die Anzahl der Bildpunkte pro Bildzeile und die Zeilenanzahl achten. Einfarbige Monitore haben den Vorteil, daß sie relativ billig sind. Die Anzeigefarbe kann Grün, Weiß und Bernstein sein. Sie können direkt an den Apple IIc angeschlossen werden.

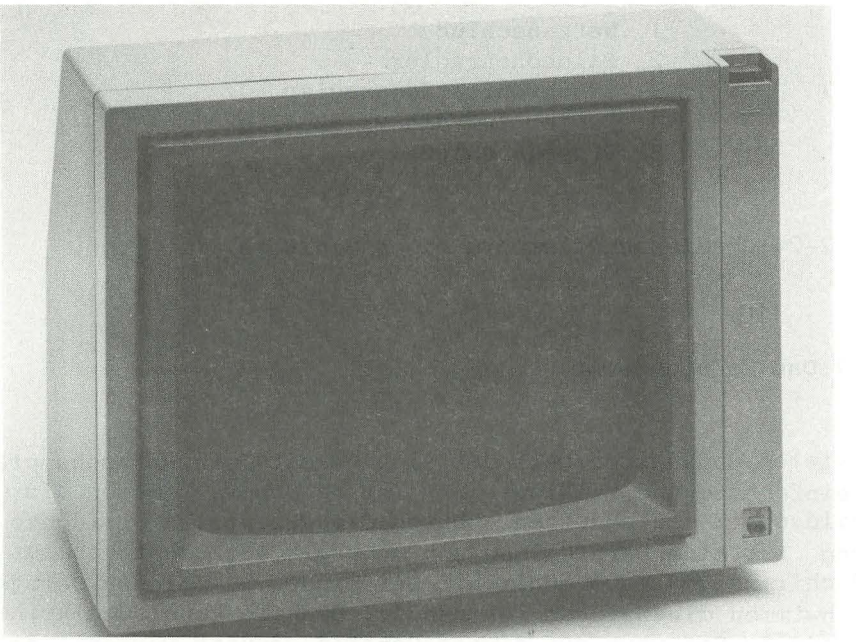
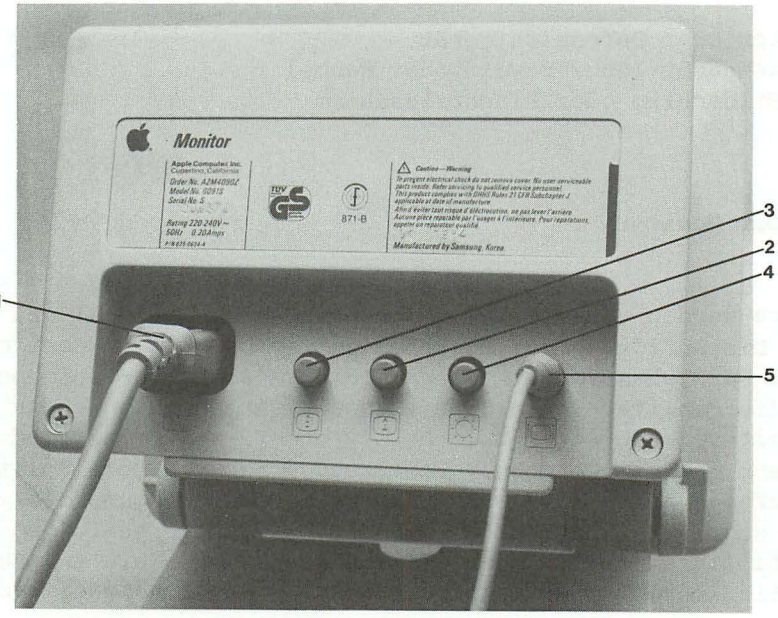


Abb. 2-23: Ein Apple-Monitor (grüne Anzeige)




1 Netzanschluß
2 Bildhöhenregler
3 Synchronisationsregler
4 Regler für die Bildhelligkeit
5 Videoanschluß

Abb. 2-24: Bedienungselemente eines Monitors

### 2.2.7 Das Fernsehgerät

In vielen Haushalten befindet sich bereits ein Fernsehempfänger. Die einfachste und billigste Art ist es daher, diesen als Computerbildschirm zu benutzen. Durch die wesentlich geringere Auflösung ist eine Darstellung von mehr als 40 Buchstaben auf einer Bildschirmzeile kaum möglich. Bei einer 80-Zeichen-Darstellung verschwimmen die Zeichen ineinander. Deshalb kann auch keine professionelle Software auf diesen Geräten genutzt werden, da diese größtenteils eine 80-Zeichen-Darstellung verlangen. Um ein Fern-

sehgerät an den Apple IIc anschließen zu können, muß der PAL-Modulator/Adapter angeschlossen werden. Die Videosignale werden über diesen Adapter in verständliche Signale für das Fernsehgerät aufbereitet.

1 Sendereinstellung
2 Lautstärke, Helligkeit, Farbe
3 Ein/Aus-Schalter

Abb. 2-25: Ein Farbfernsehgerät

### 2.2.8 Der LCD-Bildschirm

Da der Apple IIc als portabler Computer gebaut wurde, mußte auch der Bildschirm neu entwickelt werden. Dabei kam nur die LCD-Technologie (LCD = Liquid Cristal Display) in Frage. Zu deutsch: Flüssigkristallanzeige. Diese Art der Zeichendarstellung kennen Sie bestimmt auch von Quarzuhren oder digitalen Thermometern. Sie sind sehr sparsam im Energieverbrauch. Schon die Körperelektrizität reicht aus, um einzelne Segmente zum Leuchten zu bringen.

Der Flachbildschirm kann anstelle des Monitors an den Apple IIc angeschlossen werden. Das Darstellungsformat ist mit dem des Monitors identisch. So kann auch eine professionelle Textverarbeitung mit 80-Zeichen-Darstellung im portablen Betrieb genutzt werden. Als Beispiel sei hier nur Apple Works erwähnt.

Abb. 2-26: Der Apple IIc mit LCD-Flachbildschirm

## 2.3 Peripheriegeräte

Der Apple IIc ist ein in sich geschlossenes System. Es wird aber nicht lange dauern, dann werden Sie weitere Geräte an Ihren Computer anschließen wollen. Dann erst zeigen die Ausbaumöglichkeiten, was als Computersystem zu bezeichnen ist.

Wie bereits eingangs besprochen, befindet sich auf der Rückseite des Computers eine Vielzahl von verschiedenen Steckern und Buchsen. Sie sind alle zur Verbindung mit anderen Geräten gedacht. Das ermöglicht es dem Benutzer, sein System individuell auszu-

bauen. Wir wollen im folgenden Abschnitt einige dieser Erweiterungsmöglichkeiten betrachten.

### 2.3.1 Drucker

Um einen Text oder ein Listing von einem Computer ausdrucken zu können, brauchen Sie einen Drucker. Beim Apple IIc ist ein Drukkeranschluß möglich. Es gibt eine Vielzahl von verschiedenen Druckertypen. An den Anschluß des Apple IIc können aber nur Drukker angeschlossen werden, die die richtige Schnittstelle besitzen. Dafür bietet die Firma Apple entsprechende Druckermodelle an. Im Normalfall sind die Druckeranschlüsse von Computern genormt. Als Beispiel seien hier die serielle Schnittstelle RS 232 C und die parallele Centronic-Schnittstelle genannt. Es gibt aber die Möglichkeit, sich ein Kabel anfertigen zu lassen. Über eine steckbare Modulbox wird es aber bald möglich sein, parallele und serielle Schnittstellen herzustellen. Dadurch kann auch ein Drukker angeschlossen werden, der eine genormte Schnittstelle besitzt.

Die Druckermodelle, die Apple Computer liefert, können mit dem Apple IIc problemlos verbunden werden. Das Verbindungskabel wird mit einer Demodiskette und einigen Druckerhandbüchern ausgeliefert. Dabei braucht der Drucker nicht mehr an den Computer angepaßt werden.

### 2.3.2 Plotter

Oft wünscht man sich, die vom Computer erstellten Grafiken auf Papier zu bringen. Für geschäftliche oder technische Anwendungen gibt es eine Vielzahl von Programmen, die Geschäftsgrafiken (z.B. Umsatzdiagramme), Konstruktionszeichnungen und Baupläne zeichnen. Damit diese Grafiken ausgegeben werden können, verfügen die meisten dieser Programme über die Möglichkeit, einen Plotter zu unterstützen.

Der Plotter kann mit Hilfe eines Stiftes Zeichnungen von höchster Genauigkeit anfertigen. Die Genauigkeit der Geräte, liegt bei 0,05 Millimeter. Wie der Drucker kann auch der Plotter an den Apple angeschlossen werden. Er verfügt meistens über die gleichen genormten Anschlüsse wie ein Drucker.

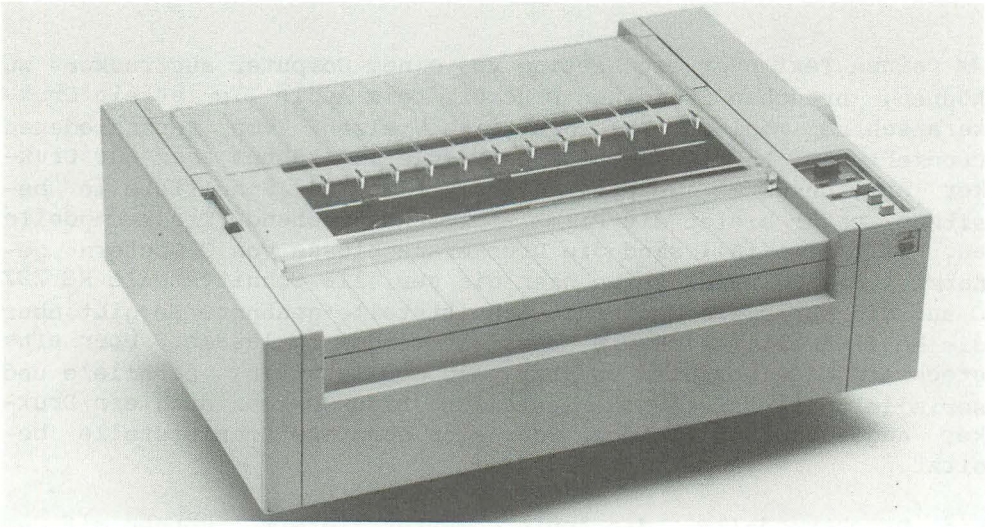


Abb. 2-27: Apple-Drucker "Imagewriter"

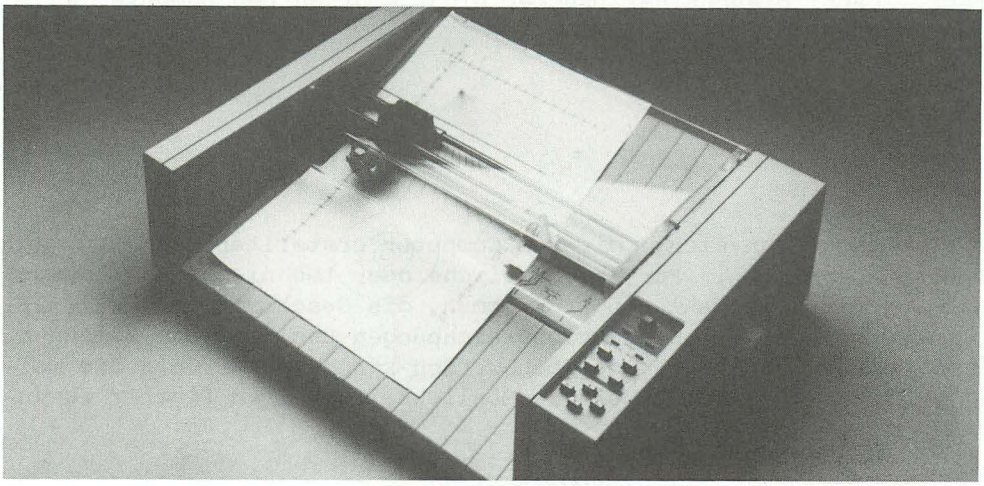


Abb. 2-28: Apple-Farbplotter Modell 410

### 2.3.3 Modem und Akustikkoppler

Die Datenfernübertragung ist ein weites Betätigungsfeld für den Computeranwender. Damit Sie mit anderen Computern Kontakt aufnehmen können, brauchen Sie ein Gerät, das die Verbindung Computer - Telefonnetz ermöglicht. Am Apple IIc ist ein Anschluß für die Datenübertragung vorgesehen, an den ein Telefonmodem angesteckt werden kann. Über eine genormte serielle Schnittstelle, die über ein Modul an den Apple angeschlossen wird, ist auch die Verbindung mit einem Akustikkoppler möglich.

Über die Anschlußbuchse, die den Rechner mit dem Modem verbinden soll, kann eine Datenübertragung von Rechner zu Rechner erfolgen. Dazu brauchen Sie kein Modem und keinen Akustikkoppler. Ein normales Übertragungskabel hilft Ihnen Daten auszutauschen. Diese Art der Datenübertragung nennt man auch Übertragung mit Null-Modem.

Abb. 2-29: Akustikkoppler

### 2.3.4 Joystick und Maus

Computerspiele sind die beste Art, die Anwendung eines Joysticks zu erklären. Der Steuerknüppel, wie der Joystick auch genannt wird, überträgt Bewegungen an den Computer. Einfache Steuerknüppel arbeiten mit vier eingebauten Schaltern, die wechselweise durch die Bewegung des Joysticks gedrückt werden, um die Richtung festzulegen. Der Apple-Joystick hat zwei Drehregler eingebaut. Das sind Drehwiderstände, deren Widerstandswerte vom Computer ermittelt werden. Bei jeder Auslenkung ändert sich der Widerstandswert und dadurch die Darstellung auf dem Bildschirm bzw. ein Wert im Arbeitsspeicher.

Die Maus hat dieselbe Aufgabe wie der Joystick. Sie wird auf einer ebenen Fläche bewegt und überträgt so eine bestimmte Position. Im Innern der Maus befinden sich zwei Übertragungsrollen und eine Andruckrolle, die von einer Kugel bewegt werden. An den Rollen befinden sich perforierte Scheiben, die über eine Lichtschranke elektronische Signale an den Computer melden.

Die jeweilige Position von Joystick oder Maus kann in BASIC-Programmen ermittelt werden. So sind Steuerungen nicht nur der Tastatur vorbehalten.

### 2.3.5 Das externe Diskettenlaufwerk

Größere Datenmengen können nur auf Datenträgern gespeichert werden, dennoch kann ein Diskettenlaufwerk auch nur eine bestimmte Menge von Daten aufnehmen. Besonders bei großen Programmen und höheren Programmiersprachen ist ein zweites Diskettenlaufwerk von großem Nutzen. Für den Apple IIc wird deshalb ein zweites externes Laufwerk angeboten. Dieses läßt sich mit wenigen Handgriffen an den Computer anschließen. Die Steuerelektronik für das zweite Laufwerk ist bereits im Computer eingebaut. So ist es möglich, ohne Diskettenwechsel auf größere Datenbestände zuzugreifen.

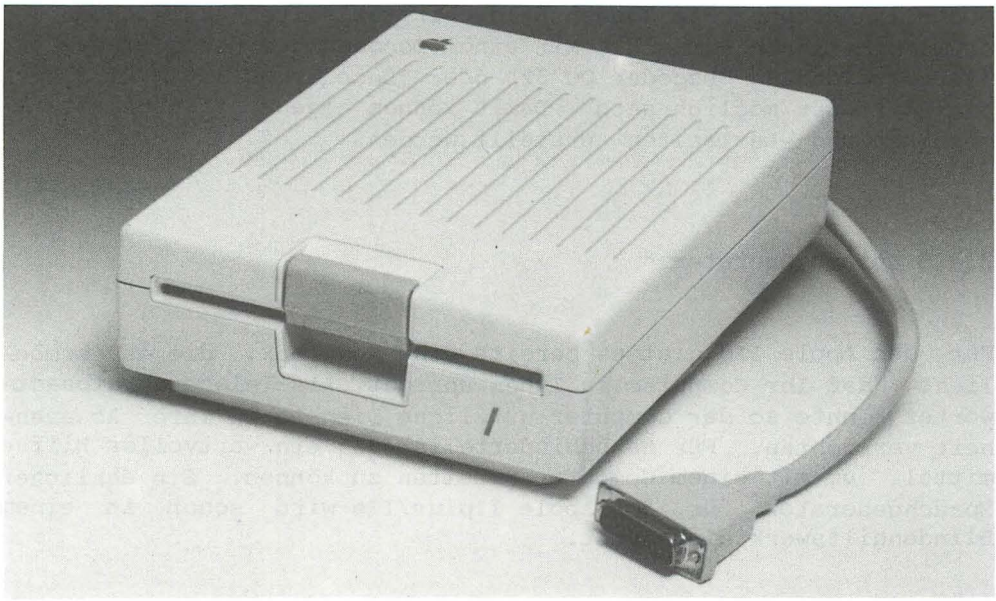


Abb. 2-30: Das externe Diskettenlaufwerk

### 2.3.6 Hard Disk/Festplattenlaufwerk

Für professionelle Anwendungen kann ein Festplattenlaufwerk nützlich sein, wenn große Datenbestände verwaltet werden müssen. Dazu zählen wissenschaftliche Anwendungen, aber auch Personalverwaltung, Buchhaltung und das Finanzwesen. Es gibt die Möglichkeit, ein Festplattensystem zwischen 5 und 65 Megabyte an den Apple IIc anzuschließen. So kann ein Datenbestand von 5 bis 65 Millionen Zeichen verwaltet werden.

### 2.3.7 Btx-Modulbox

Die Bildschirmtextbenutzung wird jetzt auch für den Apple IIc möglich. Mit einem Decoder, der über ein Interface-Kabel an den Apple angeschlossen wird, ist es möglich, den Dienst der Bundes-

post zu nutzen. Sie können Bildschirmtextseiten abspeichern und so Informationen sammeln. Mit einem Decoder der Firma Blaupunkt, Typenbezeichnung "Decoder DC 32" oder "Decoder DC 33", wird auch ein Editieren möglich sein. Damit können eigene Bildschirmtext-grafiken und -textseiten erstellt werden.

### 2.3.8 Sprachgenerator

Für den Apple IIc gibt es bereits eine Modulbox, die es ermöglicht, daß Ihr Computer zu Ihnen spricht. Als Telefonanrufbeantworter könnte so der Computer nützliche Dienste in Ihrer Abwesenheit verrichten. Für Sehbehinderte ist das ein wertvolles Hilfsmittel, um mit einem Computer arbeiten zu können. Ein ähnlicher Sprachgenerator für den Apple IIplus/IIe wird schon in einem Blindenhilfswerk eingesetzt.

### 2.3.9 Akku, Stromversorgung, Transporttasche

Die kompakte Bauweise Ihres Computers ist dazu gedacht, den Rechner portabel zu machen. Damit der Computer mit Strom versorgt werden kann, bieten verschiedene Hersteller Taschen an, die einen eingebauten Akku besitzen. Eine Akkuladung reicht etwa für 8 Stunden Betrieb. In Verbindung mit dem LCD-Bildschirm können Sie dann den Apple IIc fast überallhin mitnehmen. Außerdem dient diese Tasche zum Schutz vor Transportschäden.

Die wenigen Bauteile, die in Ihrem Rechner vorhanden sind, machen erst den portablen Einsatz möglich. Ein häufiger Diskettenzugriff verringert die Betriebsdauer des Akkus. Den Drucker oder Plotter müssen Sie leider noch zu Hause lassen.

Die Palette der Erweiterungen könnte lange fortgesetzt werden. Fragen Sie Ihren Apple-Händler, er wird Sie über Neuerungen bestimmt unterrichten. Auch Computerfachzeitschriften veröffentlichen viele Informationen und testen neue Zusatzgeräte. Wenn Sie die Augen offen halten, werden Sie Ihr Computersystem nach Ihren Wünschen auf- und ausbauen können.

# 3

# Die Bedienung des Apple IIc

## 3 Die Bedienung des Apple IIc

Bevor Sie Ihren neuen Computer bedienen, sollten Sie die Einrichtungen Ihres Rechners genau kennen. Nur so können Sie Fehlbedienungen ausschalten. Im folgenden Kapitel werden wir gemeinsam den Apple IIc aufstellen und die Peripheriegeräte anschließen. Auch wenn Sie Ihren Computer selbst ohne jede Hilfe aufgebaut haben, sollten Sie dennoch diesen Abschnitt genau durchlesen. Gerade die fehlerhafte Installation kann den Spaß am Programmieren schnell verderben.

### 3.1 Aufstellung und Inbetriebnahme

Beim Aufstellen Ihres Computers müssen Sie immer darauf achten, daß alle Geräte richtig verbunden werden. Der Standort muß so gewählt werden, daß der Computer vor äußeren Einflüssen geschützt ist. Das bedeutet:

- keinen Standort wählen, der den ganzen Tag der Sonneneinstrahlung ausgesetzt ist.

- Der Computer soll auf einer ebenen, glatten Fläche erschütterungsfrei aufgestellt werden.

- Auch wenn der Apple IIc tragbar ist - er ist doch ein empfindliches elektronisches Gerät (besonders das Diskettenlaufwerk).

Sie sollten daher den Computer immer sorgfältig behandeln. Er wird es Ihnen durch lange Lebensdauer danken. Wenn Sie ihn transportieren wollen, so empfehle ich Ihnen, die Orginalverpackung zu benutzen.

Auf der Rückseite Ihres Apple IIc befindet sich der Tragegriff. Er muß vor der Inbetriebnahme ausgeklappt werden, damit eine Luftzirkulation im Gerät stattfinden kann. Abbildung 3-1 zeigt uns, wie wir den Computer aufstellen.

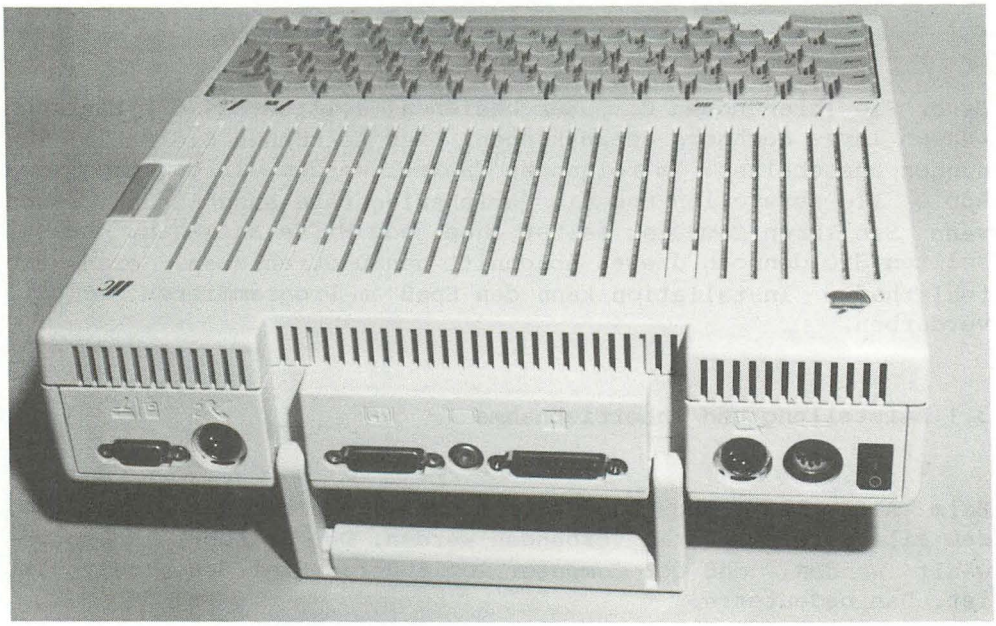


Abb. 3-1: Der Tragegriff muß ausgeklappt sein!

## 3.2 Verbindung mit dem Netz

Die Reihenfolge bei der Aufstellung sollten Sie auf jeden Fall einhalten. Wenn der Rechner aufgestellt ist, muß das Netzteil installiert werden. Um das interne Schaltnetzteil vor Induktionsspannungen zu schützen, sollten Sie zunächst den Transformatorblock mit dem Computer verbinden. Das steckbare Kabel können Sie anschließend mit dem Netz verbinden.

**ACHTUNG! Stellen Sie sicher, daß der Computer ausgeschaltet ist!**

Die Abbildungen 3-2 und 3-3 sollen Ihnen beim Anschluß Ihres Computers an das Netz helfen.

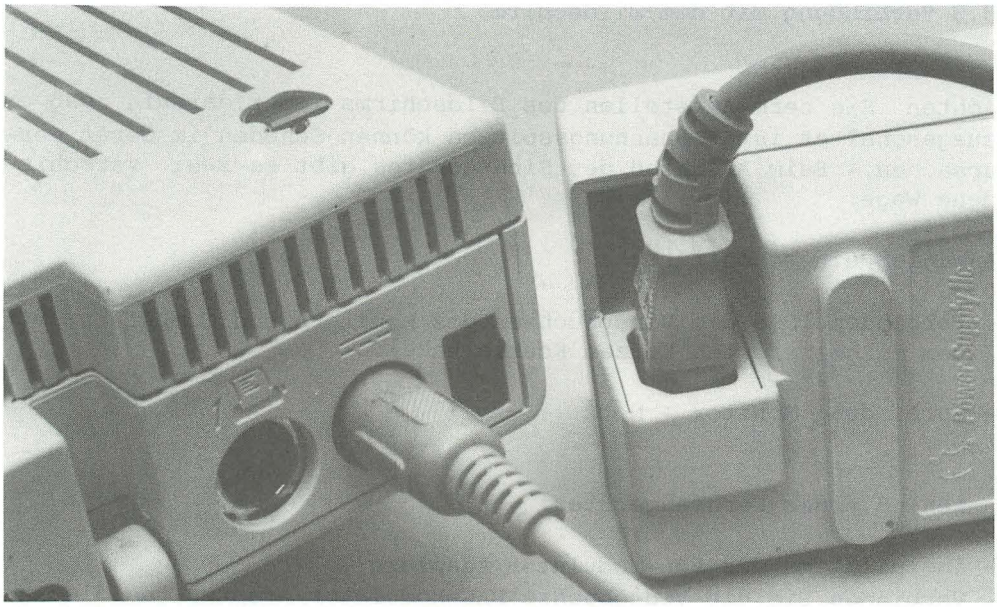



Abb. 3-2: Netzverbindung mit dem Computer

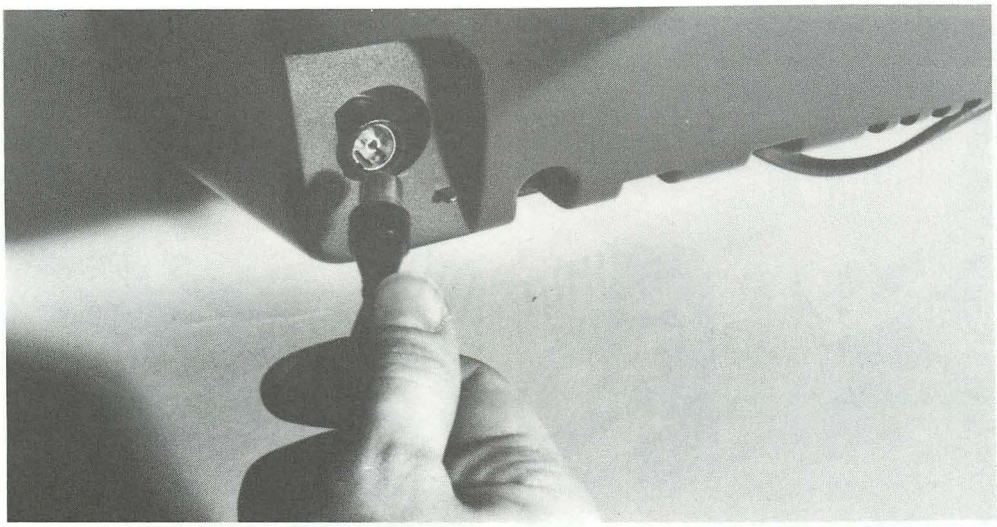


Abb. 3-3: Verbindung mit dem Bildschirm

## 3.3 Verbindung mit dem Bildschirm

Achten Sie beim Aufstellen des Bildschirms immer darauf, daß er ausgeschaltet ist. Spannungsspitzen können Schäden im Gerät verursachen. Beim Anschluß des Sichtgerätes gibt es zwei verschiedene Wege:

Anschluß eines Monitors:

- Verbinden Sie die Videobuchsen des Monitors und Computers mit dem mitgelieferten kurzen Koaxialkabel.

- Verbindung Monitor - Netz

Anschluß eines Fernsehgerätes:

- Installation des PAL-Modulator/Adapters
- Verbinden Sie die Videobuchse PAL-Modulator/Adapter und die Antennenbuchse des Fernsehgeräts mit dem beigepackten langen Koaxialkabel.
- Verbindung Fernsehgerät - Netz

Abb. 3-4: Fernsehgeräteanschluß

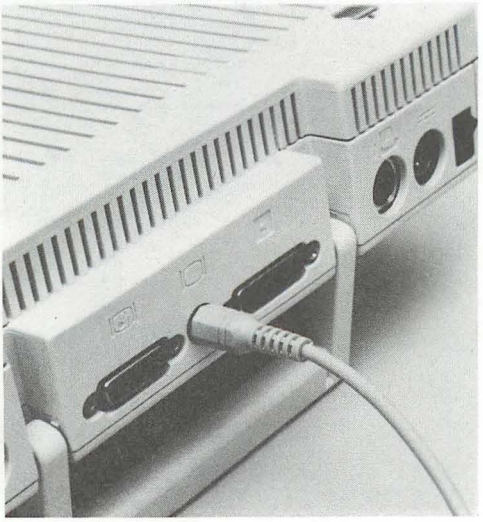


Abb. 3-5: Monitoranschluß

## 3.4 Einschalten, Abstimmung mit dem Bildschirm

Bevor Sie den Computer einschalten, überprüfen Sie die Anschlüsse auf Richtigkeit und festen Sitz. Das Diskettenlaufwerk sollte vor dem Einschalten des Computers geöffnet werden.

Folgende Reihenfolge sollte eingehalten werden:

1. Einschalten des Monitors oder Fersehgerätes
2. Einschalten der Peripherie (Drucker, Plotter)
3. Diskettenlaufwerk öffnen
4. Computer einschalten

## 3.5 Systemmeldung

Nach dem Einschalten des Computers muß die grüne Betriebslampe leuchten. Das Diskettenlaufwerk beginnt zu laufen. Jetzt sollten Sie die jeweilige Diskette einschieben. Wenn keine Diskette im Laufwerk vorgefunden wird, erscheint die Meldung:

**Check Disk Drive**

Am oberen Bildschirmrand gibt der Computer die Meldung aus:

**Apple IIc**

Ist eine Diskette eingelegt, lädt der Computer das STARTUP-Programm. Das kann von Diskette zu Diskette unterschiedlich aussehen.

## 3.6 Wenn nichts passiert!

Sie sollten nicht gleich einen Schreck bekommen, wenn der Computer einmal nicht das tut, was er tun soll. Meist liegt ein kleiner Fehler vor, der leicht behoben werden kann.

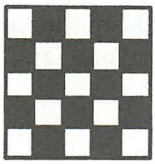
Folgende Fehler können auftreten:

- Computer ist nicht an das Netz angeschlossen
- Netzstecker steckt nicht richtig in seiner Buchse
- Der Bildschirm ist nicht angeschlossen
- Bildschirm ist nicht eingeschaltet
- Bild ist auf Dunkel gestellt

Überprüfen Sie in diesem Fall nochmals die Anschlüsse. Wenn wirklich nichts funktioniert, gehen Sie zu Ihrem nächsten Apple-Händler und lassen das Gerät überprüfen.

## 3.7 Der Cursor

Der Cursor oder Zeiger, wie wir ihn auch nennen können, hat die Aufgabe, dem Benutzer zu zeigen, wo er sich gerade auf dem Bildschirm befindet. Er kann die Zeilen entlanggleiten, er kann aber auch von Zeile zu Zeile springen. Man kann mit ihm löschen und auch einfügen. Er blinkt, damit man sieht, wo er sich befindet. Das Aussehen des Cursors wird vom geladenen Programm bestimmt. Er kann verschiedene Formen annehmen:



Blinkendes Quadrat

Cursor mit Kreuz

Voll ausgezeichnetes Quadrat

Blinkender Strich

Die Cursorsteuertasten lassen den Cursor auf dem Bildschirm wandern. Auch mit Hilfe der CONTROL-Taste kann der Cursor bewegt werden.

## 3.8 Bildschirmformat

Im eingeschalteten Zustand ist die Textanzeige des Bildschirms immer auf 40 Zeichen eingestellt. Die Ausnahme ist beim Starten eines Programms, das die Zeichendarstellung softwaremäßig auf 80 Zeichen umschaltet. In der Betriebsart BASIC kann mit der ESC-Taste und den Zifferntasten 4 oder 8 von 40 auf 80 Zeichen umgeschaltet werden. Folgende Vorgehensweisen bieten sich an:

80-Zeichen-Darstellung wählen:

- Drücken der ESC-Taste (Der Cursor wird invers mit einem stehenden Kreuz)
- Anschließend Betätigen der Taste 8

40-Zeichen-Darstellung wählen:

- Drücken der ESC-Taste
- Betätigen der Taste 4

Besonders bei der Programmierung in BASIC kann die 80 Zeichendarstellung von Bedeutung sein. Beim Auflisten der Programme wird das Listing übersichtlicher, da die langen BASIC-Zeilen doppelt soviel Platz benötigen. Es können auf dem Bildschirm bis zu 24 Zeilen für Text dargestellt werden.

## 3.9 Die Tastatur

Bei der Tastatur des Apple IIc ist einiges zu beachten. Es gibt die Möglichkeit, vom deutschen Zeichensatz auf US-Zeichensatz umzuschalten. Dabei verlieren jedoch einige Tasten ihre gültige Beschriftung. Andererseits können einige Programme ohne bestimmte US-Zeichen nicht arbeiten. Als Beispiel sei hier Apple-LOGO erwähnt.

Die Umlaute (Ä,ä,Ö,ö,Ü,ü), das ß und das Paragraphzeichen (§) werden beim Umschalten zu:

| | | | |
|---|---|---|---|
| Ä | = [ | Ü | = ] |
| ä | = { | ü | = } |
| Ö | = \ | ß | = ~ |
| ö | = \| | § | = @ |

Auch die Zeichen und Z und Y haben andere (vertauschte) Tastenbelegungen. Dies wird im Benutzerhandbuch verschwiegen, was beim Programmieren Fehler hervorrufen kann.

Sie sollten auf jeden Fall die deutsche Tastenbelegung verwenden, solange das programmtechnisch möglich ist. Wird die US-Tastenbelegung häufiger gebraucht, ist anzuraten, daß Sie die fehlende Beschriftung mit Selbstklebeetiketten oder wischfesten Stiften anbringen. Es kann aber auch eine Tabelle der Belegungen auf den freien Platz oberhalb der Tastatur geklebt werden. Sie müssen auf jeden Fall beachten, daß englische Software mit der US-Tastatur bedient wird.

Abb. 3-6: Die deutsche Tastenbelegung

Abb. 3-7: Die US-Tastenbelegung

## 3.10 Kompatibilität zu anderen Apple-Computern

Der Apple IIc ist eine Weiterentwicklung der Apple-II-Serie. Darum mußte auch die Kompatibilität gewahrt bleiben. Software, die für den Apple II, Apple IIplus und den Apple IIe geschrieben ist, läuft grundsätzlich auch auf dem Apple IIc. Es gibt einige Unterschiede, die eventuell Schwierigkeiten bereiten können. Wenn die Software stark in den Arbeitsspeicher eingreift und interne Monitorroutinen benötigt, so kann es sein, daß die Programme leicht modifiziert werden müssen.

Da der eingebaute Mikroprozessor einige Befehle und Adressierungsarten hinzubekommen hat, kann es unter Umständen zu Überschneidungen mit bestimmten Befehlssequenzen kommen. Bevor Sie also Software für Ihren Apple IIc kaufen, sollten Sie darauf achten, daß diese für den Apple IIc tauglich ist.

Programme, die eine Erweiterungskarte benötigen, sind auf dem IIc nicht lauffähig. Sie müssen an die Hardware des IIc angepaßt werden. Programme, die die 80-Zeichen-Karte ansprechen laufen ohne Bedenken. Es wird für Apple IIc bald eine Vielzahl angepaßter Software geben, da das Betriebssystem DOS 3.3 durch ProDOS ersetzt wird. Dazu bieten Fremdhersteller parallel Hardwarezusätze in großer Zahl an.

### Apple-III-Software auf dem IIc

Disketten des Apple III können im Apple IIc dann gelesen werden, wenn der Apple III unter dem SOS-Betriebssystem gelaufen ist und der Apple IIc mit ProDOS geladen wurde. SOS ist das Standard-Betriebssystem des Apple III und mit ProDOS verwandt. Die Datenstruktur dieser Betriebssysteme ist sehr ähnlich. So können angelegte Daten des Apple III mit dem IIc ausgetauscht werden. Die Software des Apple III wird jedoch auf dem IIc nicht lauffähig sein.

# 4

# BASIC auf dem Apple IIc

## 4 BASIC auf dem Apple IIc

Im Laufe der letzten Jahre hat sich BASIC als Standardprogrammiersprache bei vielen Heim- und Personalcomputern durchgesetzt. BASIC ist die Abkürzung von Beginners All-Purpose Symbolic Instruction Code, was soviel bedeutet wie: Universelle Befehlssprache für Anfänger.

Seit dem Jahr 1964, als die erste BASIC-Version in der Universität in Dartmouth (USA) entwickelt wurde, haben sich viele BASIC-Dialekte gebildet. Der Apple IIc spricht zwei dieser BASIC-Dialekte: Applesoft-BASIC und Integer-BASIC.

Beim Einschalten des Computers steht Ihnen sofort Applesoft-BASIC zur Verfügung. Integer-BASIC kann von Diskette aus in den Rechner geladen werden. Obwohl beide Programme BASIC-Dialekte darstellen, kann doch das Programm des einen nicht unter der Verwaltung des anderen laufen und umgekehrt.

Normalerweise wird Applesoft-BASIC für den Apple IIc verwendet. Bei einigen Anwendungen kann es dennoch von Nutzen sein, wenn mit Integer-BASIC gearbeitet wird. Die genaueren Zusammenhänge und Unterschiede beider BASIC-Versionen werden Ihnen in diesem Kapitel aufgezeigt.

Beim Einschalten Ihres Computers wird das Betriebssystem von der Diskette geladen. Dabei erscheint, je nach Inhalt des geladenen Programms, das Anfangsbild, oder Ihr Computer springt direkt in den BASIC-Interpreter. Sie können aber auch mit BASIC arbeiten, ohne eine Diskette einzulegen.

Das Programm Applesoft-BASIC ist im Computer fest installiert. Es braucht nicht von der Diskette geladen werden. Integer-BASIC muß eingelesen werden. Es befindet sich auf der System-Masterdiskette DOS 3.3. Integer-BASIC wird bei dieser Diskette automatisch in den Speicherbereich des Computers geladen. Diese Diskette wird mit dem Apple IIc nicht ausgeliefert. Sie können sie aber bei jedem Apple-Händler erwerben. Auch kann Integer-BASIC nicht unter ProDOS betrieben werden, da Ihnen dazu ein Integer-Systemloader zur Verfügung stehen müßte.

# 4 BASIC auf dem Apple IIc

Nach dem Einschalten des Apple IIc leuchtet die rote Signallampe auf und zeigt den Betrieb des Diskettenlaufwerks an. Nach mehreren Sekunden erscheint bei nicht eingelegter Diskette:

**Apple IIc**

und etwas weiter unten auf dem Bildschirm:

**Check Disk Drive.**

Ihr Computer erwartet das Einlegen einer Diskette, um ein Programm laden zu können. Wenn Sie nun die CONTROL-Taste und gleichzeitig die RESET-Taste drücken, springt Ihr Computer in den Programmodus Applesoft-BASIC. Sie können nun in Applesoft-BASIC programmieren, sind aber in diesem Moment nicht in der Lage, BASIC-Programme abzuspeichern. Der Grund liegt darin, daß Sie kein Betriebssystem geladen haben, womit Ihr Rechner Diskettenverwaltungsaufgaben übernehmen kann. Dieses Programm muß sich aber im internen Arbeitsspeicher befinden, um Daten auf die Diskette übertragen zu können.

## BASIC-Startdisketten

Um ein Abspeichern Ihrer Programme zu gewährleisten, müssen Sie eine Startdiskette erstellen, mit der Sie eines der Betriebssysteme laden können. Sie haben in BASIC die Auswahl zwischen zwei dieser Betriebssysteme, deren Arbeitsweise im Moment unwichtig ist und Sie nur verwirren würde. Im Kapitel 8 und 9 werden Sie genaueres darüber erfahren. Im Moment sollte für uns nur das Abspeichern und Laden von Wichtigkeit sein, um längere Programme nicht immer wieder neu eintippen zu müssen.

Wir unterscheiden zwei dieser Betriebssysteme: DOS und ProDOS.

DOS ist die Abkürzung von Disk Operating System, was soviel bedeutet wie Diskettenverwaltungsprogramm. Das gleiche gilt auch für ProDOS, nur mit dem Zusatz Pro für Professional.

## 4.1 BASIC-Startdiskette unter ProDOS

Eine Startdiskette für BASIC unter ProDOS erstellen wir wie folgt:

Schalten Sie Ihren Computer aus, und legen Sie die Diskette "System-Dienstprogramme" für den IIc in das interne Laufwerk ein. Nach dem Einschalten und einer kurzen Ladezeit erscheint das Hauptmenü wie in Abbildung 4-1 dargestellt. Wählen Sie bitte den Programmpunkt "6.FORMATIEREN EINER DISKETTE", indem Sie die Taste mit der Zahl 6 drücken. Sie können aber auch mit den Cursorsteuertasten den gewünschten Programmpunkt anwählen und mit der RETURN-Taste die Auswahl bestätigen.

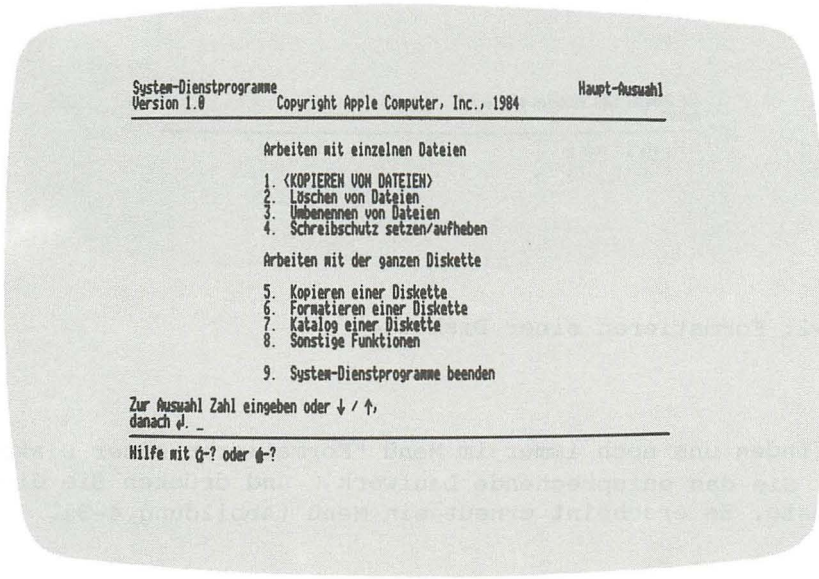



Abb. 4-1: Menü System-Dienstprogramme

Nun zeigen zwei Pfeile den Abschnitt "Formatieren einer Diskette" an. Nach dem Drücken der RETURN-Taste erscheint das Menü wie in Abbildung 4-2. Der Computer fragt nach dem Laufwerk, in welchem sich die Diskette befindet, die formatiert werden soll. Ist ein zweites Laufwerk vorhanden, kann die Systemdiskette im internen Laufwerk belassen werden. Die Diskette, die formatiert wird, liegt dann im zweiten Laufwerk. Wenn kein externes Diskettenlaufwerk vorhanden ist, müssen die Disketten entsprechend den Anweisungen des Programms gewechselt werden.

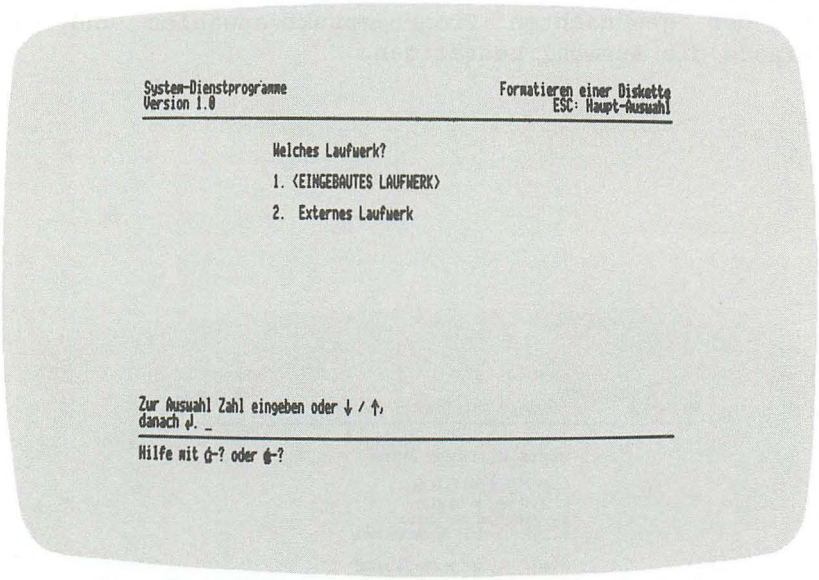



Abb. 4-2: Formatieren einer Diskette

Wir befinden uns noch immer im Menü "Formatieren einer Diskette". Wählen Sie das entsprechende Laufwerk, und drücken Sie die RETURN-Taste. Es erscheint erneut ein Menü (Abbildung 4-3).

```
System-Dienstprogramme                    Formatieren einer Diskette
Version 1.0                                          ESC: Haupt-Auswahl

                    Verwendet wird: Eingebautes Laufwerk

                    Wählen Sie das Format:

                    1. <PRODOS>

                    2.  DOS3.3

                    3.  Pascal

                    4.  Ich weiß nicht


Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵. _

Hilfe mit ⌂-? oder ⌘-?
```

Abb. 4-3: Formatabfrage

```
System-Dienstprogramme                    Formatieren einer Diskette
Version 1.0                                          ESC: Haupt-Auswahl

                    Verwendet wird: Eingebautes Laufwerk

                    Format: ProDOS

                    Neuer Disketten-Name: /DISK27

          Diese Funktion löscht /UTILITIES?   <JA>  Nein


J(a) oder N(ein), mit ← / → auswählen,
danach ↵. _
```

Abb. 4-4: Daten werden gelöscht!

Da wir eine Startdiskette unter ProDOS erstellen wollen, wählen wir den Menüpunkt "1.PRODOS". Durch das Drücken der RETURN-Taste meldet das Programm, daß wir die Diskette, die formatiert werden muß, einlegen sollen. Nach erneutem Drücken der RETURN-Taste warnt uns das Programm, wie in Abbildung 4-4 dargestellt.

ACHTUNG! Nach Betätigen der RETURN-Taste werden alle Daten auf der eingelegten Diskette gelöscht, auch solche, die geschützt sind. Sie sollten auf jeden Fall sicherstellen, daß die Diskette neu oder leer ist. Bei beschriebenen Disketten ist darauf zu achten, daß die Daten nicht mehr gebraucht werden.

Drücken Sie nun bitte die RETURN-Taste, um den Formatierungsvorgang einzuleiten. Das Laufwerk rattert, und die Diskette im Laufwerk beginnt zu rotieren. Lassen Sie sich durch dieses Geräusch nicht irritieren, es ist ganz normal. Die Diskette wird dabei in Bereiche eingeteilt, die für den späteren Diskettenbetrieb wichtig sind. Was dabei im einzelnen geschieht, lesen Sie im Kapitel 8 noch genauer.

Es können nun weitere ProDOS-Disketten formatiert werden, oder Sie gehen zum Hauptmenü zurück. Nach Beenden der Formatierung wählen Sie im Haupmenü den Punkt "1.KOPIEREN VON DATEIEN". Daß diese Eingabe mit RETURN beendet wird, muß wohl nicht mehr gesagt werden. Das Programm fragt Sie zuerst nach dem Laufwerk, in dem sich die Originaldiskette befindet. Danach muß das Laufwerk angegeben werden, in dem sich die Zieldiskette befindet.

Wie in Abbildung 4-5 zu sehen ist, fragt der Computer, ob wir alle Dateien oder nur einige kopieren wollen. Bitte wählen Sie "EINIGE" aus. Es werden nun alle Dateinamen von der Originaldiskette in den Computer eingelesen, damit Sie die entsprechende Datei aussuchen können.

```
System-Dienstprogramme                          Kopieren von Dateien
Version 1.0                                     ESC: Haupt-Auswahl

                    Original-Diskette: Eingebautes Laufwerk

                    Ziel-Diskette: Eingebautes Laufwerk

                    Einige oder Alle Dateien bearbeiten? (EINIGE) Alle

E(inige) oder A(lle), mit ← / → auswählen,
danach ↵. _

Hilfe mit ⇧-? oder ⇧-?
```

Abb. 4-5: Dateienabfrage

Nun müssen wir die Dateinamen PRODOS und BASIC.SYSTEM mit der Cursorsteuertaste "Pfeil nach rechts" markieren. Die Dateinamen können mit den Cursortasten "Pfeil nach oben" und "Pfeil nach unten" ausgewählt werden. Sollten Sie einen falschen Dateinamen markiert haben, ist der Fehler mit Betätigen der Cursorsteuertaste "Pfeil nach links" zu beheben. Drücken Sie die RETURN-Taste, um die ausgewählten Dateien zu kopieren.

Nach Beendigung des Kopiervorgangs wählen Sie im Hauptmenü den Punkt "9.System-Dienstprogramme beenden". Quittieren Sie mit RETURN die Antwort "JA". Der Computer springt in den Modus Applesoft-BASIC und befindet sich jetzt im Betriebssystem ProDOS, da beim Einlesen der System-Dienstprogrammdiskette ProDOS geladen wurde. Es könnten jetzt ProDOS-Anweisungen gegeben werden. Die kopierte Diskette kann als BASIC-Startdiskette unter ProDOS genutzt werden.

## 4.2 BASIC-Startdiskette unter DOS 3.3

Die System-Masterdiskette DOS 3.3 wird nicht mit dem Apple IIc ausgeliefert. Darum müssen wir uns mit einem kleinen Trick behelfen. Wir benötigen dazu die Diskette "Spaß mit dem Apple", die im Lieferumfang Ihres Computers enthalten ist.

Wenn Sie das Diskettenlabel (Beschriftungsetikett) genauer betrachten, werden Sie die Aufschrift DOS 3.3 erkennen. Genau dieses Betriebssystem benötigen wir, um eine Startdiskette unter DOS 3.3 zu erstellen. Sie können auch eine DOS-3.3-Systemmaster-Diskette oder eine Diskette benutzen, die im DOS-3.3-Format beschrieben ist.

Haben Sie den Apple IIc bereits eingeschaltet, schieben Sie die Diskette "Spaß mit dem Apple" in das Diskettenlaufwerk. Die Diskette starten Sie mit dem gleichzeitigen Drücken der RESET-, CONTROL- und OFFENER-APFEL-Taste und anschließendem Loslassen der CONTROL-Taste. Bei ausgeschaltetem Gerät startet die Diskette beim Einschalten automatisch.

Ist das Programm geladen, erscheint auf dem Bildschirm ein Menü, das acht Programmpunkte anzeigt. Wählen Sie den Programmpunkt "8.Ende". Dies geschieht mit den Pfeiltasten oder nach Eingabe der Programmnummer. Ist der Programmpunkt ausgewählt, wird das Programm nach der Eingabe von RETURN abgebrochen und veranlaßt Sie, eine neue Diskette einzulegen. Dabei wird der Bildschirm gelöscht.

Sie sehen auf dem Bildschirm den Cursor, der blinkend hinter einem großen Ü oder einer eckigen Klammer steht. Der Computer zeigt Ihnen an, daß er sich in Applesoft-BASIC befindet. Legen Sie eine neue oder leere Diskette in den Laufwerksschacht, und schließen Sie das Diskettenlaufwerk.

Wir wollen ein kleines Programm schreiben, um die Startdiskette benennen zu können. Geben Sie dazu folgende Anweisungen ein, und beenden Sie jede Programmzeile mit RETURN.

```
NEW
10 HOME
20 PRINT "DOS 3.3"
30 END
```

Jetzt folgt die Anweisung an den Computer, daß die Diskette im Laufwerk mit dem DOS-Format beschrieben wird. Die Diskette ist danach in Spuren und Sektoren eingeteilt. Geben Sie dazu folgende Befehle ein:

INIT HELLO

Das Laufwerk wird kurzzeitig rattern und surren. Danach beginnt es, die Diskette zu initialisieren (einzuteilen). Was dabei genau vorgeht, können Sie im 8. Kapitel nachlesen.

Ist der Vorgang beendet, haben Sie eine Startdiskette für Applesoft-BASIC unter DOS 3.3 erstellt. Sie können das durch einen Neustart überprüfen. Auch nach dem Starten der DOS-3.3 Systemmaster-Diskette wird der gleiche Vorgang ablaufen. Nach dem Starten der Disketten wurde das Betriebssystem DOS 3.3 geladen. Wir haben nun die Voraussetzung, um mit dem Diskettenlaufwerk unter DOS 3.3 arbeiten zu können.

**BASIC-Versionen**

Applesoft-BASIC und Integer-BASIC sind zwei der BASIC-Versionen, die von Apple direkt angeboten werden. Es gibt aber noch einige Fremdhersteller, die Applesoft-BASIC-Erweiterungen anbieten. Mit diesen Programmzusätzen ist es möglich, noch weitere Programmierhilfen anzuwenden. Wir werden uns einige Unterschiede zwischen Applesoft- und Integer-BASIC jetzt ansehen.

## 4.3 Integer-BASIC

Wenn Sie den Apple IIc ohne zusätzliche Software gekauft haben, steht Integer-BASIC nur dann zur Verfügung, wenn Sie sich die DOS-3.3-System-Master-Diskette kaufen. Auf dieser Diskette befindet sich die Integer-BASIC-Datei.

Integer-BASIC heißt im Deutschen Ganzzahl-BASIC. Wie wir schon am Namen sehen, können wir damit nur ganze Zahlen verarbeiten. Sie werden sich nun fragen, welche Vorteile das bringen soll. Ganz

einfach! Integer-BASIC ist gegenüber Applesoft-BASIC um einiges schneller, wenn es um komplexe Berechnungen geht. Durch das Fehlen der Nachkommastellen erhöht sich die Verarbeitungsgeschwindigkeit.

Eine automatische Zeilennumerierung ist nur mit Integer-BASIC möglich. Jedoch kann dieses wiederum keine Dezimalbrüche verarbeiten, wie es Applesoft-BASIC bietet.

Dennoch haben Integer- und Applesoft-BASIC einiges gemeisam, was wir auf den folgenden Seiten durcharbeiten wollen.

## 4.4 Applesoft-BASIC

Wenn Sie Ihren Apple IIc einschalten, so steht Ihnen sofort Applesoft-BASIC zur Verfügung. Dieses BASIC gestattet Ihnen das Programmieren von komplizierten mathematischen Problemen. Es ist deshalb besonders bei der Erstellung aufwendiger Grafiken von Vorteil. Diese Vorteile müssen Sie sich durch eine langsamere Verarbeitungsgeschwindigkeit erkaufen. Sie werden aber in diesem Kapitel einiges über die strukturierte und rationelle Programmierung erfahren. Dadurch können Sie schon bei der Programmierung die Verarbeitungsgeschwindigkeit Ihrer Applesoft-BASIC-Programme beeinflussen.

Auf den folgenden Seiten werden alle Eigenschaften von Applesoft-BASIC genauestens erklärt. Die Unterschiede zu Ganzzahl- oder Integer-BASIC können Sie anhand einfacher Beispiele genau mitverfolgen.

## 4.5 Programmieren in BASIC

Um in BASIC programmieren und anschließend speichern zu können, haben wir uns schon eine Startdiskette erstellt. Wenn Sie die Möglichkeiten von Integer-BASIC ausnutzen wollen, müssen Sie die DOS-3.3-System Master-Diskette auf Ihrem Apple IIc starten. Dadurch wird automatisch Integer-BASIC in den Speicherbereich Ihres

Computers geladen. Nach dem Starten der Diskette meldet sich der Computer mit:

**APPLE II**
**DOS VERSION 3.3 SYSTEM MASTER**

**JANUARY 1, 1983**

**LOADING INTEGER BASIC**
**INTO MEMORY.**

anschließend mit:

**COPYRIGHT APPLE COMPUTER,INC. 1980,1982**

**BE SURE CAPS LOCK IS DOWN**

Durch die Meldung LOADING INTEGER BASIC INTO MEMORY signalisiert der Computer, daß er Integer-BASIC in den Speicher geladen hat. Danach werden Sie zum Drücken der CAPS-LOCK-Taste aufgefordert. Die Großschreibung der Anweisungen ist beim Apple IIc nur unter DOS 3.3 für DOS-Befehle und einige Systembefehle erforderlich.

Nach dem Laden der DOS-3.3-System-Master-Diskette erscheint je nach Schalterstellung des Tastaturwahlschalters unter der Meldung BE SURE CAPS LOCK IS DOWN eine großes Ü oder eine eckige Klammer. Der Computer zeigt so die Bereitschaft zur Aufnahme von Anweisungen an.

## 4.6 Umschalten von Applesoft- auf Integer-BASIC

Um auf Integer-BASIC umzuschalten, müssen Sie folgenden Befehl in Großbuchstaben eingeben:

INT

Der Computer zeigt jetzt ein Größer-als-Zeichen (>) an. Er befindet sich im BASIC-Modus Integer-BASIC. Dadurch wurden alle Programmzeilen in Applesoft-BASIC, die sich noch im Speicher befunden haben, gelöscht. Integer-Befehle können nur dann in den Computer eingegeben werden, wenn dieses Zeichen am linken Bildschirmrand den Modus anzeigt. Der Befehl INT darf nur zum Einschalten des Integer-BASIC verwendet werden. Beim Programmieren darf er auf keinen Fall angewandt werden, da er das eingegebene Programm löschen würde.

Um von Integer-BASIC wieder in Applesoft-BASIC zu kommen, müssen Sie den Befehl

FP

in Großbuchstaben eingeben. Dabei werden alle eingegebenen Programmzeilen in Integer-BASIC gelöscht!

## 4.7 Was bei der Tastatur zu beachten ist

Wenn Sie das 3. Kapitel gelesen haben, dann wissen Sie, daß es zwei verschiedene Belegungen bei der Tastatur des Apple IIc gibt. Dies läßt sich durch den Schalter rechts neben der RESET-Taste realisieren, der mit Quadraten symbolisch gekennzeichnet ist. Nichtgedrückt läßt er die Bearbeitung mit dem deutschen Zeichensatz zu. Die Beschriftung der Tasten stimmt mit der Belegung überein.

Wenn Sie den Schalter in gedrückter Stellung benutzen, haben Sie die US-Tastenbelegung, wobei die Beschriftung nicht mehr übereinstimmt. Die Abbildungen 4-6 und 4-7 zeigen die Unterschiede.

Abb. 4-6: Deutsche Tastenbelegung

Abb. 4-7: US-Tastenbelegung

Wenn Sie nach der Eingabe von Programmzeilen auf eine der Tastenbelegungen umschalten, so ändern sich einige Zeichen auf dem Bildschirm. Da der Computer nur eine gewisse Anzahl von Zeichencodes versteht, interpretiert er diese in einem anderen Modus als die Zeichen, die normalerweise für diesen Code vorgesehen sind. Sie müssen daher darauf achten, daß Sie diese Zeichen möglichst nicht benutzen, wenn Sie zwischen den Zeichensätzen umschalten wollen. In der Tabelle im Abschnitt 3.10 sehen Sie die Zeichen, die sich beim Umschalten verändern. Tippen Sie diese in den Computer ein, und betätigen Sie die Tastaturumschaltung. Es kann dabei nichts im Programm oder im Computer zerstört werden. Die Zeileninhalte bleiben durch das Umschalten unverändert und können im Normalzustand weiterverarbeitet werden. Da alle Zeichen, die Sie in BASIC benötigen, auf der deutschen Tastenbelegung vorrätig sind, sollten Sie auch diese benützen, um eine Fehlbedienung auszuschließen.

## 4.8 BASIC-Betriebsarten

In BASIC unterscheiden wir zwischen zwei Betriebsarten. Die erste ist die Betriebsart Direktanweisung. Anweisungen werden direkt an den Computer gegeben und sofort von ihm ausgeführt.

Die zweite Betriebsart nennen wir Programmieren. Der Computer wird auf die Abarbeitung von Anweisungen vorbereitet. Diese Arbeit verrichtet er nach Eingabe eines bestimmten Startbefehls.

Der Computer benötigt ein Programm, um seine Arbeiten zu erledigen. Auch der Applesoft-BASIC-Interpreter besteht aus solch einem Programm. Der Rechner hat also einen integrierten Wortschatz, den wir uns zunutze machen wollen. Bei der Abarbeitung einer Anweisung vergleicht der Computer die Angaben auf dem Bildschirm mit den Wortmustern in seinem Speicher und folgerichtig mit den programmierten Mustern im BASIC-Interpreter.

BASIC-Interpreter deshalb, weil das integrierte Programm die BASIC-Befehle interpretiert und nicht übersetzt. Das BASIC-Programm wird nicht in Maschinensprache umgewandelt. Dazu würden Sie einen BASIC-Compiler benötigen.

Wenn die Anweisungen mit den Wortmustern übereinstimmen, wird die Anweisung ausgeführt. Es kommt aber oft vor, daß sie nicht richtig sind. Der Computer zeigt das durch eine Fehlermeldung an. Die Meldung lautet:

**SYNTAX ERROR** **in Applesoft-BASIC**

***** SYNTAX ERROR** **in Integer-BASIC**

Der Syntax ist die Schreibweise oder der Satzaufbau einer BASIC-Anweisung. Ist dieser Aufbau falsch, so zeigt der Computer uns an, daß wir eine fehlerhafte Eingabe gemacht haben. Der Apple IIc kennt noch andere Fehlermeldungen, die wir noch besprechen werden.

### 4.8.1 Die Direktanweisung

Nach dem Einschalten des Apple IIc und dem Starten der BASIC-Startdiskette befindet sich der Apple im Modus "Direktanweisung". Es sei denn, der Computer hat bereits ein Programm gestartet. In diesem Modus können Sie dem Computer einzelne Befehle eingeben und auf ihre Richtigkeit überprüfen. Er unterscheidet sich vom Programmiermodus dadurch, daß keine Programmzeilennummern vergeben werden. Die Eingabe wird vom Computer sofort ausgeführt.

Im folgenden Abschnitt ist es ohne Bedeutung, ob Sie sich im Applesoft-BASIC oder im Integer-BASIC befinden. Schreibweise und Ausführung sind in beiden BASIC-Versionen gleich.

Wenn Sie auf einem Taschenrechner eine Addition ausführen, so zeigt Ihnen der Rechner das Ergebnis sofort nach der Eingabe der Zahlenwerte, dem Pluszeichen und dem Ist-gleich-Zeichen an. Bei unserem Computer ist das nicht ganz so leicht. Wir müssen ihm noch mitteilen, daß er uns das Ergebnis auch anzeigen soll. Dazu haben wir den Befehl

**PRINT**

PRINT heißt drucken, was noch aus der Zeit stammt, als Drucker als Ausgabegeräte benutzt wurden. Dieser Befehl dient zur Ausgabe von Zeichen auf dem Bildschirm oder einem Drucker. In der Betriebsart Direktanweisung können wir ihn zum Beispiel zur Ausgabe von Texten und Zahlen verwenden. Tippen Sie folgendes in den Computer ein:

```
PRINT "Apple IIc"
```

Nach dem Drücken der RETURN-Taste (die Taste mit dem abgewinkelten Pfeil), sollten Sie folgendes Ergebnis auf Ihrem Bildschirm sehen:

```
ÜPRINT"Apple IIc"
Apple IIc
```

Das was auf dem Bildschirm erscheint, muß in Anführungszeichen gesetzt werden. Dabei kann der Befehl PRINT in Applesoft-BASIC auch klein geschrieben werden. Der Integer-BASIC-Interpreter verlangt eine Eingabe des Befehls in Großbuchstaben. Die Buchstaben in Anführungszeichen können Sie in Klein- und Großschrift eingeben.

Der Bildschirm kann im 40- und 80-Zeichen-Modus in BASIC benutzt werden. Eine Darstellung der BASIC-Zeilen mit 80 Zeichen ist für die Übersichtlichkeit von Vorteil.

Wenn Sie an der letzten Position in einer Zeile angekommen sind, springt der Cursor auf die nächste Zeile. Dabei ist zu beachten, daß in Applesoft-BASIC 255 Zeichen in einer Anweisungszeile stehen dürfen. Beim Erreichen der höchstmöglichen Zeichen ertönt ein Signalton, der Sie warnt. Haben Sie die 255 Zeichen überschritten, so gibt der Computer ein großes Ö oder einen Schrägstrich aus. Dabei wird die Eingabe aus dem Arbeitsspeicher gelöscht. In Integer-BASIC sind es bei einer PRINT-Anweisung 120 Zeichen. Wenn die Höchstzahl erreicht ist, wird bei der Ausführung die Fehlermeldung

```
*** TOO LONG ERR
```

ausgegeben. Die Zeile wird ebenfalls aus dem Arbeitsspeicher gelöscht. Sie muß gekürzt und neu eingeben werden.

In der Betriebsart Direktanweisung können in Applesoft-BASIC auch mehrere Befehle in einer Zeile stehen. Integer-BASIC läßt dies nicht zu. Eine einzelne PRINT-Anweisung wird mit einem Zeilenvorschub ausgeführt. Zum Beispiel:

```
ÜPRINT"Apple Computer":PRINT"Apple Computer"
Apple Computer
Apple Computer


>PRINT"Apple Computer":PRINT"Apple Computer"
*** SYNTAX ERR


Üprint"Huber Max":PRINT:PRINT"Maier Franz"
Huber Max

Maier Franz


>print"Alles Gute zum Geburtstag!"
*** SYNTAX ERR

ÜPRINT"Print:PRINT:print"

Print:PRINT:print
```

Sie sehen an diesen Beispielen, welche Möglichkeiten Sie haben, den Befehl PRINT zum Schreiben auf dem Bildschirm anzuwenden. Der Befehl PRINT bleibt in Anführungszeichen unberücksichtigt. Anführungszeichen in einer PRINT-Anweisung können eine unbeabsichtigte Darstellung zur Folge haben.

Wenn mehrere Anweisungen in einer Zeile ausgeführt werden sollen, so werden diese durch einen Doppelpunkt (:) getrennt. Der Doppelpunkt bewirkt nach jeder Anweisung einen Zeilenvorschub. Es gibt noch einige weitere PRINT-Anwendungen, die wir im Abschnitt "Programmieren" anschauen.

Nun wollen wir aber wieder auf das Beispiel mit dem Taschenrechner zurückkommen. Um auf Ihrem Apple IIc zu rechnen, müssen Sie einige Zeichen kennenlernen, die der Computer anders als auf einem Blatt Papier darstellt. Wir können in der Betriebsart Direktanweisung addieren, subtrahieren, multiplizieren, dividieren und

potenzieren. Auch andere Rechenfunktionen können ausgeführt werden, doch diese Möglichkeiten wollen wir beim Programmieren ausprobieren. Folgende Zeichen müssen Sie verwenden:

| | |
|---|---|
| + | **für die Addition** |
| - | **für die Subtraktion** |
| * | **für die Multiplikation** |
| / | **für die Division** |
| ^ | **für das Potenzieren** |

Damit der Computer auch ein Ergebnis ausgibt, wird vor jeder Rechnung eine PRINT-Anweisung gesetzt. In Integer-BASIC dürfen Sie keine Dezimalstellen verwenden. Die Zahlen müssen im Bereich zwischen -32767 und +32767 liegen, denn Integer-BASIC kann nur Zahlen in diesem Bereich verarbeiten. Wenn diese Zahlen über- oder unterschritten werden, erfolgt die Fehlermeldung

***** >32767 ERR**

Zwischen den einzelnen Zeichen können Leerschritte eingegeben werden, diese werden, sofern sie nicht in Anführungszeichen stehen, nicht berücksichtigt. Die Beispiele sollen das erläutern. Die BASIC-Version erkennen Sie am Bedienerführungszeichen.

```
ÜPRINT 3+7
10

ÜPRINT 23 + 37
60

ÜPRINT 8 - 13
-5

ÜPRINT 12 * 5
60

ÜPRINT 78/12
6.5
```

In Applesoft-BASIC können Sie die Anweisung PRINT auch mit dem Fragezeichen (?) abkürzen.

```
Ü? 23 + 7 + 67
97

Ü? 5 ^ 4
625

>PRINT 12/5
2

>PRINT 45 * 34
1530

>PRINT 20000 + 12767
32767

>PRINT 20000 + 12767 + 1
***   32767 ERR

>PRINT 12 ^ 2
144

>? 24 + 6
*** SYNTAX ERR
```

Im Integer-BASIC werden Dezimalbrüche auf- oder abgerundet. Applesoft-BASIC berechnet Dezimalzahlen mit bis zu 9 Dezimalstellen. Was darüber liegt, wird auf- oder abgerundet. Dezimalstellen werden durch einen Dezimalpunkt gekennzeichnet. Er entspricht dem Komma, das nicht verarbeitet wird. Bei mehr als 9 Dezimalstellen wird die Exponentenschreibweise angewendet.

```
Ü? 12.0834567935 + 4
16.0834568

>PRINT 23/34
0

ÜPRINT 23/34
.676470588

>PRINT -12 + -34
-46
```

```
Ü? 789456123789 + 1
7.89456124E+11

Üprint 12*12*12/7*12-34/23
257.378882

ÜPRINNT 12+12

?SYNTAX ERROR
```

In der Betriebsart Direktanweisung müssen Sie nach jeder Eingabe mit der RETURN-Taste dem Computer mitteilen, daß er die Anweisungen abarbeiten soll. Wenn mehrere Anweisungen in einer Zeile stehen, können sie bereits ein kleines Programm darstellen. Diese Möglichkeit ist nur in Applesoft-BASIC gegeben. Geben Sie folgende Befehlssequenz ein:

```
HOME:PRINT"Mein erstes Programm!":PRINT:
FOR X=1 TO 100:PRINT "TOLL";:NEXT
```

Nach der Eingabe von RETURN erscheint nachstehende Ausgabe:

```
Mein erstes Programm!

TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL
! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TO
LL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL!
TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL
! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TO
LL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL!
TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL
! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TO
LL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL!
TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL
! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TO
LL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL!
TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL
! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL! TO
LL! TOLL! TOLL! TOLL! TOLL! TOLL! TOLL!
Ü
```

Sie haben gesehen, daß es mit wenigen, einfachen Befehlen möglich ist, lauffähige Anweisungen zu schreiben. Direktanweisungen wer-

den uns bei der Programmierung in BASIC immer wieder begegnen. Sie helfen uns bei der Erstellung von Programmen, bei der Datenverwaltung und Diskettenbenutzung. Um aber die Möglichkeiten von BASIC voll ausnutzen zu können, bedarf es einer Programmstruktur, die wir im folgenden Abschnitt genau betrachten wollen.

### 4.8.2 Programmieren

In der Betriebsart Direktanweisung ist es uns nicht möglich, längere Programme zu schreiben. Für den praktischen Einsatz eignet sich daher diese Art der Eingabe von Anweisungen nur sehr begrenzt. Nach der Eingabe von RETURN wurde die Zeile sofort ausgeführt. Um aber Programme schreiben zu können, bedarf es eines Hilfsmittels, das es uns ermöglicht, diese in beliebiger Länge einzugeben.

Die Betriebsart Programmieren stellt uns diese Möglichkeiten zur Verfügung. Die eingegebene Programmzeile wird nach der Eingabe von RETURN im Speicher des Computers abgelegt. Im Speicherbereich des Apple IIc ist ein Speicherplatz für BASIC-Programme vorgesehen.

BASIC-Programme sind aus einzelnen Programmzeilen aufgebaut. Jede dieser Zeilen besitzt eine Programmzeilennummer. In Applesoft-BASIC können diese Zeilennummern zwischen 0 und 63999 liegen. Es dürfen nur ganze Zahlen verwendet werden. Wird der Höchstbereich überschritten, so erfolgt die Fehlermeldung SYNTAX ERROR. Integer-BASIC erlaubt Zahlen zwischen 0 und 32767. Bei einer Überschreitung erscheint die Fehlermeldung *** >32767 ERR.

Bevor Sie mit dem Programmieren beginnen, sollten Sie den Bildschirm löschen, um den Überblick zu behalten. Dazu verwenden Sie bitte die Anweisung

**HOME**

in Applesoft-BASIC. Diese Anweisung löscht den Bildschirm und läßt den Cursor in die obere linke Ecke des Bildschirms springen. Die eingegebenen Programmzeilen bleiben im Speicher erhalten. Um auch den Speicherbereich in BASIC zu löschen, geben Sie den Befehl

**NEW**

Damit werden alle im BASIC-Speicher befindlichen Programmzeilen gelöscht.

**VORSICHT!** Die Anweisung NEW kann ein mühsam erstelltes Programm in Bruchteilen von Sekunden löschen. Speichern Sie deshalb längere Programme in regelmäßigen Zeitabständen auf Diskette.

Nun haben wir den Computer für die Eingabe der ersten Programmzeilen vorbereitet. Geben Sie bitte folgende Zeilen ein:

```
10 PRINT 12 * 123
20 PRINT 1 + 1
30 PRINT 8 - 5
```

Nach der Eingabe von RETURN hat sich gar nichts getan. Richtig! Der Computer hat diese Zeile in seinem internen Speicher abgelegt. Wenn wir den Bildschirm mit HOME löschen, bleibt die Eingabe dennoch erhalten. Um zu sehen, was wir im BASIC-Speicher abgelegt haben, können wir das mit der Anweisung

**LIST**

wieder sichtbar machen. Auf dem Bildschirm erscheint folgende Meldung:

```
ÜLIST
 10  PRINT 12 * 123
 20  PRINT 1 + 1
 30  PRINT 8 - 5
Ü
```

Was wir hier sehen, ist bereits ein Programm. Dieses Programm umfaßt drei Programmzeilen. Sie müssen dem Computer nun mitteilen, daß er das Programm ausführen soll. Mit

**RUN**

geben Sie die Anweisung der Programmausführung. Bestätigen Sie mit RETURN.

In den folgenden Abschnitten werde ich die Eingabe von RETURN bei Direktanweisungen und dem Abschluß einer Programmzeile voraussetzen. Die Anweisung RUN bewirkt, daß der Computer beginnt, die Programmzeile auszuführen.

Unser Beispiel nach RUN:

```
276
2
3
```

Die Zeilennummern spielen dabei eine wichtige Rolle. Sie zeigen dem Computer den richtigen Weg durch das Programm. In Integer-BASIC müssen Sie auch das Ende eines Programms angeben. Dazu verwenden Sie die Anweisung

**END**

Wenn dieser Befehl fehlt, erscheint die Fehlermeldung

***** NO END ERR**

In Applesoft-BASIC kann dieser Befehl entfallen. Der Applesoft-Interpreter bricht das Programm nach der letzten Programmzeile automatisch ab.

Zum Testen eines längeren BASIC-Programms ist der Befehl gut geeignet. An bestimmten Stellen eingesetzt ist es möglich, Programmteile aus dem Hauptprogramm herauszulösen. Nach einer erfüllten Bedingung kann z.B. IF A = 0 THEN END folgen.

## 4.9 Die Programmzeilennumerierung

Der BASIC-Interpreter nutzt die Programmzeilennummern als Wegweiser durch das Programm. Jede Programmzeile hat eine Art Hausnummer, die diese kennzeichnet. Wenn Sie zweimal die gleiche Zeilennummer vergeben, wird nach der Eingabe von RETURN die alte Zeile überschrieben. Es gilt aber immer nur die zuletzt eingegebene Zeile. Sie sehen das an folgendem Beispiel:

```
ÜNEW
Ü10 PRINT "Haus"
Ü10 PRINT "Auto"

ÜRUN
Auto
```

Im Gegensatz zu der Betriebsart Direktanweisung wird beim Programmieren die Eingabe gespeichert. Bei Vergabe einer Zeilennummer am Anfang einer Programmzeile wird der Programmiermodus eingeleitet. Nach dem Quittieren mit RETURN speichert der Computer die Programmzeile im BASIC-Speicherbereich ab. Wenn Sie den Bildschirm mit der Anweisung HOME löschen, können Sie das Programm trotzdem starten. Es wurde vom Computer abgespeichert.

Die Programmzeilen müssen nicht nacheinander eingegeben werden. Sie können die Programmzeilen auch durcheinander eintippen. Nach dem Auflisten werden Sie sehen, daß der BASIC-Interpreter die Zeilennummern geordnet hat. So können Sie auch im nachhinein Zeilen einfügen.

```
Ü10 PRINT "ROT"
Ü20 PRINT "BLAU"
Ü12 PRINT "GELB"
Ü40 PRINT "BRAUN"

ÜLIST

10  PRINT "ROT"
12  PRINT "GELB"
20  PRINT "BLAU"
40  PRINT "BRAUN"
ÜRUN
```

```
ROT
GELB
BLAU
BRAUN
```

Wenn Sie genau aufgepaßt haben, werden Sie die Zeilennumerierung in Zehnerschritten bemerkt haben. Diese Art der Numerierung sollten Sie auch anwenden. So können Sie jederzeit neue Programmzeilen einfügen, wenn das Programm geändert wird. Als Faustregel gilt eine Schrittweite von Zehn. Wissen Sie schon am Anfang Ihrer Arbeit, welchen Umfang das Programm haben wird, so müssen Sie es für zusätzliche Programmteile in größere Schritte aufteilen.

## 4.10 Anzeigen der Programmzeilen

Den Befehl LIST benötigen wir zum Zeigen der erstellten Programmzeilen. Nun kann es aber vorkommen, daß Sie schon ein längeres Programm eingegeben haben, als der Bildschirm zeigen kann. Wenn wir jetzt LIST eingeben, rollt der Bildschirm die Zeilen wie auf einer Papierrolle ab. Bei längeren Programmen müssen die ersten Zeilen den nachfolgenden weichen. Es kann also immer nur ein gewisser Ausschnitt gezeigt werden. Um aber auch die anderen Programmzeilen wieder sichtbar zu machen, können wir eine Art Bremse einbauen. Wenn Sie beim Auflisten die CONTROL-Taste und die Taste C gleichzeitig drücken, hält der Computer an und meldet BREAK, was soviel bedeutet wie unterbrechen. Sie können das Listing anhalten und fortsetzen mit den Tasten CONTROL und S. Wenn bestimmte Programmzeilen aufgelistet werden sollen, gibt es einige zusätzliche Zeichen im Befehl LIST, die dies bewirken. Geben Sie die folgenden Zeilen ein:

```
10 PRINT "10"
20 PRINT "20"
30 PRINT "30"
40 PRINT "40"
50 PRINT "50"
60 PRINT "60"
```

Sie wollen alle Zeilen zwischen 20 und 50 aufgelistet haben. Dazu können Sie folgende Anweisung geben.

```
Eingabe: LIST 20,40 (RETURN)

Ergebnis: 20 PRINT "20"
          30 PRINT "30"
          40 PRINT "40"
          50 PRINT "50"
```

Es kann auch von einer vorgewählten Programmzeilennummer bis zum Programmende aufgelistet werden.

```
Eingabe: LIST 40,

Ergebnis: 40 PRINT "40"
          50 PRINT "50"
          60 PRINT "60"
```

Wenn Sie vom Anfang des Programms bis zu einer bestimmten Zeilennummer listen wollen, müssen Sie folgendes eingeben.

```
Eingabe: LIST ,40

Ergebnis: 10 PRINT "10"
          20 PRINT "20"
          30 PRINT "30"
          40 PRINT "40"
```

Sie sehen, daß es viele Möglichkeiten gibt, Ihr erstelltes Programm zu begutachten.

**AUTO, MAN**

In Integer-BASIC haben Sie die Möglichkeit, die Programmzeilennummern automatisch zu vergeben. Dazu verwenden Sie bitte den Befehl AUTO. Sie können diese Anweisung nur in Integer-BASIC anwenden. Zum Befehl AUTO muß noch die erste Programmzeilennummer mit angegeben werden. Wir wollen das an folgenden Beispielen ausprobieren.

```
>AUTO 10

>10 ....
>20 ....
>30 ....
>40 ....
```

```
>AUTO 25

>25 ....
>35 ....
>45 ....
>55 ....
```

Die Schrittweite beträgt in der Regel immer 10. Es können aber auch andere Schrittweiten als die Standardweite (10) eingestellt werden. Die erste Zahl zeigt die Startzeile an, die zweite Zahl die Schrittweite:

```
>AUTO 100,100

>100 .....
>200 ......
>300 .......

>AUTO 124, 12

>124 ....
>136 .....
>148 ......
```

Die Programmzeile kann mit der **CONTROL**-Taste und der Taste **X** gelöscht werden. Die Zeilenautomatik wird mit dem Befehl MAN wieder ausgeschaltet. So können Sie wieder, wie gewohnt, Zeilennummern vergeben.

## 4.11 Anmerkungen in Programmzeilen

Beim Programmieren in BASIC sollten Sie immer auf eine gute Gliederung und Dokumentation Ihrer Programme achten. Nur so können Sie später Änderungen vornehmen und sich im Programm zurechtfinden.

Um die einzelnen Programmteile kennzeichnen zu können, müssen Anmerkungen im Programm eingefügt werden. Dazu gibt es die Anweisung **REM**. Diese Bemerkungen werden bei der Ausführung des Programms nicht berücksichtigt. Dennoch können sie auch Teil einer

aktiven BASIC-Zeile sein. Das folgende Beispiel soll einige Arten der Dokumentation zeigen.

```
Ü10 PRINT " " : REM Das ist ein Leerzeichen
Ü20 PRINT "$" : REM Das ist das Dollarzeichen
Ü30 REM *** Multiplikation ***
Ü40 PRINT 10 * 12
Ü50 END

RUN


$
120

LIST

10 PRINT " " : REM Das ist ein Leerzeichen
20 PRINT "$" : REM Das ist das Dollarzeichen
30 REM *** Multiplikation ***
40 PRINT 10 * 12
50 END
```

Wenn Sie Bemerkungen in das Programm eingefügt haben, können diese Programmzeilen später als Sprungzeilennummern für Unterprogramme benutzt werden. In diesem Fall müssen Sie die REM-Anweisung im Programm belassen, da sonst der Programmablauf unterbrochen wird.

## 4.12 Editieren des Programms

Bei der Erstellung eines neuen Programms kommt es immer wieder vor, daß Fehler entstehen. Da wird ein Zeichen zuviel eingetippt, oder es fehlt eine Programmzeile im Programm. Um diese Mißgeschicke zu beheben, haben wir einen Editor zur Verfügung. Als Editor bezeichnen wir die Möglichkeit, mit Hilfe des Computers Änderungen im Programm vornehmen zu können.

### 4.12.1 Zeichen einfügen und verändern

In vielen Fällen kommt es vor, daß einzelne Zeichen zu schnell eingetippt wurden und darum falsch sind. Diese Fehler kann man durch einfaches Überschreiben des falschen Zeichens beheben. Wir müssen dazu nur den Cursor um so viele Stellen nach links rücken, bis wir das fehlerhafte Zeichen mit dem Cursor überdecken. Dazu verwenden wir die Pfeil-Taste "Pfeil nach links". Diesen Vorgang können wir aber nur dann benutzen, wenn die Programmzeile noch nicht mit RETURN abgeschlossen ist.

Sollte die Programmzeile mit RETURN bereits abgeschlossen sein, oder Sie entdecken in einem bereits gelisteten Programm einen Fehler, so gibt es mehrere Möglichkeiten, ihn zu beheben.

Da wäre zunächst das Neueintippen der Programmzeile. Besonders bei großen und langen Fehlern ist diese Möglichkeit anzuraten. Bei der Eingabe der alten Zeilennummer wird diese automatisch nach Eingabe von RETURN überschrieben.

Dann gibt es den Editiercursor. Wenn Sie die ESC-Taste drücken, wandelt sich der Cursor in ein stehendes Quadrat mit einem Kreuz um. Sie befinden sich nun im Editiermodus. Solange der Cursor diese Form besitzt, kann in den Programmzeilen nichts überschrieben werden. Sie können also beruhigt mit den Cursorsteuertasten auf dem Bildschirm wandern.

Um eine Programmzeile zu ändern, muß diese ab der ersten Zahl der Programmzeilennummer mit dem Cursor markiert werden. Dazu müssen Sie den Cursor durch erneutes Drücken der ESC-Taste in den Eingabemodus bringen. Beim Überfahren der Programmzeile mit dem Cursor sollten nun die fehlerhaften Stellen geändert werden. Dieser Vorgang stellt für den Computer dasselbe dar, wie wenn Sie die Zeichen nochmals neu eingegeben hätten. Am Ende der Programmzeilen muß die RETURN-Taste betätigt werden, um die Eingabe zu bestätigen.

Bei längeren Programmen ist anzuraten, daß Sie die fehlerhafte Programmzeile nochmals neu auflisten. So haben Sie die Zeile übersichtlich am unteren Bildschirmrand und können diese einfach editieren. Sie sollten aber darauf achten, daß die Programmzeile die zulässige Zeichenlänge nicht überschreitet. Wenn Sie beim Überfahren der Programmzeile vom normalen Weg mit dem Cursor ab-

wandern, so können sich dadurch weitere Programmfehler einschleichen. Meistens aber nimmt der Computer solche Programmzeilen überhaupt nicht an.

### 4.12.2 Löschen einzelner Zeichen/ganzer Zeilen

Wenn Sie Zeichen in einer Zeile löschen wollen, müssen Sie diese mit Leerzeichen überschreiben. Dabei gelten dieselben Regeln wie bei der Änderung von Zeichen und Zeilen.

Programmzeilen können auf zwei Arten gelöscht werden. Sie können nur die Zeilennummer der Programmzeile eingeben, die gelöscht werden soll. Dann wird die Programmzeile nicht mehr beim Auflisten erscheinen. Außerdem kann, wenn der Cursor in der Programmzeile steht, diese mit der Tastensequenz CTRL-Taste und Taste X gelöscht werden. Wir wollen uns das an folgendem Beispiel anschauen.

Eingabe:

```
10 REM Das ist ein
20 REM Beispiel
30 REM wie Programmzeilen
40 REM gelöscht werden
```

```
LIST
```

Ausgabe auf dem Bildschirm:

```
10 REM Das ist ein
20 REM Beispiel
30 REM wie Programmzeilen
40 REM gelöscht werden
```

Eingabe:

40 (RETURN)

LIST

Ausgabe:

```
10 REM Das ist ein
20 REM Beispiel
30 REM wie Programmzeilen
```

Eingabe:

```
40 REM gelöscht werden
50 REM Test Test Test (CTRL-X)
```

Ausgabe:

(In Zeile 50) Ö (Am Ende der Zeile 50)

```
10 REM Das ist ein
20 REM Beispiel
30 REM wie Programmzeilen
40 REM gelöscht werden
```

Mit dieser Art des Zeilenlöschens wird es sehr schwierig sein, mehrere Zeilen oder ganze Programmteile zu löschen. Dazu gibt es einen eigenen Befehl, der auch nur eine Zeile löschen kann. Mit der Anweisung

**DEL**

kann ein bestimmter Programmteil definiert werden, der zu löschen ist. Die Zahlenkombination hinter DEL gibt die Anfang- und Endzeilennummer, getrennt durch ein Komma, an.

Beispiel:

```
10 REM
20 REM
30 REM
40 REM
50 REM
60 REM
70 REM
80 REM
90 REM
```

Mit DEL 70,90 werden die Zeilen 70 bis 90 gelöscht

Mit DEL 0,20 werden die Zeilen ab 0 bis 20 gelöscht

Mit DEL 30,70 sind alle Zeilen von einschließlich 30 bis 70 gelöscht

Werden Zeilennummern eingegeben, die nicht vorhanden sind, ignoriert sie der BASIC-Interpreter und löscht bis zur angegebenen Stelle. Er löscht auch dort, wo keine Programmzeilen vorhanden sind.

### 4.12.3 Bildschirm löschen

Um den Bildschirm zu löschen, haben wir die Anweisung HOME. Sie bewirkt das Löschen der Anzeige und das Wandern des Cursor in die linke, obere Ecke. Sie können so immer den Überblick über die Eingaben behalten.

### 4.12.4 Programmzeilen einfügen

Immer wieder kommt es vor, daß Programme geändert werden. Das Einfügen von Programmzeilen ist dabei die häufigste Arbeit. Wie schon bei der Programmzeilennumerierung erwähnt, sollten die Zeilennummern nicht zu eng beieinander liegen. Durch einfaches Einfügen von Zeilen, deren Nummer noch nicht vergeben ist, kann der Abstand vergrößert werden. Dabei müssen Sie darauf achten, daß schon vorhandene Zeilennummern nicht doppelt verwendet werden. Die alte Programmzeile wäre sonst verloren. Dazu ein Beispiel:

```
Programm:  10 REM
           20 REM
           30 REM

Einfügen:  15 REM
           16 REM

Neues Programm:  10 REM
                 15 REM
                 16 REM
                 20 REM
                 30 REM
```

## 4.13 Zahlen, Texte und Variable

In unseren BASIC-Programmen müssen wir immer unterscheiden zwischen Zahlen, Texten und Variablen, denn viele BASIC-Anweisungen verlangen eine genaue Definition. Sie sollten dem Computer mitteilen, um welche Art von Daten es sich handelt. Um den Unterschied deutlich zu machen, wollen wir uns die verschiedenen Datentypen etwas genauer anschauen.

### Zahlen

Zahlen oder numerische Werte begegnen uns in allen Bereichen des täglichen Lebens. Sie dienen dem Menschen zum Darstellen von Mengen oder Werten. Dabei unterscheiden wir zwei Arten von Zahlen:

- ganze Zahlen
- Dezimalzahlen

### 4.13.1 Ganze Zahlen

In Integer-BASIC benutzen wir nur diese Art von Zahlen. Sie dürfen keine Dezimalstellen aufweisen, können aber positiv oder negativ sein. Dabei zeigt das Vorzeichen an, ob es sich um eine positive (z.B. 456) oder um eine negative (z.B. -456) Zahl handelt. Zahlen ohne Vorzeichen werden automatisch als positive Zahlen betrachtet. Ganze Zahlen sind:

```
100
-2
31467
0
-30278
```

### 4.13.2 Dezimalzahlen

Dezimalzahlen, auch Gleitkommazahlen genannt, sind positive oder negative Zahlen, die Kommastellen besitzen dürfen. Das heißt, auch ganze Zahlen können Dezimalzahlen sein. Diese Zahlen können aber nur in Applesoft-BASIC angewendet werden. Dabei müssen wir beachten, daß das Kommazeichen in BASIC als Punkt dargestellt wird, entgegen der deutschen Schreibweise, dem Komma. Bis zu neun Stellen werden dargestellt, alle größeren Zahlen werden vom Computer in die Exponentenschreibweise umgewandelt. Zum Beispiel:

```
          0
     -31287
      54678
      0.224
     -0.567
   45689.987
   -0.00007
3 E + 02 = 300
-567895672 =  -5.67895672E+08
3456789543  = 3.45678954E+09
```

Wenn nicht alle Dezimalstellen ausgerechnet werden können, dann rundet der Computer nach der neunten Stelle auf oder ab. Folgende Beispiele sollen das verdeutlichen:

```
ÜPRINT -9876543219
-9.87654322E+09

ÜPRINT 987123654987213
9.987123655E+14

ÜPRINT 12.999999999
13

ÜPRINT -12.999999999
-13
```

### 4.13.3 Texte

Alle Zeichen, die wir über die Tastatur in den Computer eingeben können, werden als alphanumerische Daten bezeichnet. Diese Daten können auch Anweisungen oder Zeichencodes sein, die wir nicht auf dem Bildschirm sehen können. Sie werden auch als Characterstrings (Zeichenketten) bezeichnet. Bei der Eingabe von Zahlen, Buchstaben und Zeichen werden die Signale, die von der Tastatur kommen, in einen Code umgewandelt. Dieser Code hat die Bezeichnung ASCII, das ist die Abkürzung von American Standart Code for Information Interchange; genormter Code für die Zeichendarstellung in Computern. In Applesoft-BASIC können Texte bis zu 255 Zeichen enthalten. Dabei werden Leerzeichen und "nicht druckende Zeichen" mitgezählt (z.B. Controlzeichen).

### 4.13.4 Variable

Ein Programm bekommt erst dann einen Sinn, wenn es universell einsetzbar ist. Um es so flexibel wie möglich zu machen, benötigen wir Variable. Aber was sind Variable? Wir können sie auch als Platzhalter bezeichnen. Jeder Zahlenwert und Textteil der im Programm abgeändert werden muß, bekommt einen solchen Platzhalter

zugeteilt. Es gibt in Integer-BASIC und Applesoft-BASIC drei Arten von Variablen:

- Numerische Variable
- Textvariable
- Ganzzahl-Variable

Es ist Ihnen sicher aufgefallen, daß wir hier wieder unseren drei Datentypen begegnen. Diese drei Variablenarten werden aber in den zwei Basic-Versionen unterschiedlich behandelt. Als numerische Variable bezeichnen wir all die Variablen, die Zahlen darstellen. Texte und Sonderzeichen, aber auch Zahlen, werden in Textvariable zusammengefaßt. Ganzzahl-Variable enthalten, wie der Name schon sagt, ganze Zahlen.

### 4.13.5. Numerische Variable

Bei einer Berechnung, die wir über die PRINT-Anweisung ausgeben, müssen wir den Rechengang immer wieder neu eintippen. Um dies zu vermeiden, benutzen wir numerische Variable. Am folgenden Beispiel wollen wir gleich mehrere BASIC-Anweisungen kennenlernen.

**Die Aufgabe:** Es sollen vier Zahlenwerte addiert werden. Der Computer fragt die vier Zahlenwerte ab und gibt das Ergebnis aus. Wir geben folgende Programmzeilen in Applesoft-BASIC ein:

Eingabe: 10 HOME
(Diese Programmzeile löscht den Bildschirm)

Eingabe: 20 PRINT "Addition von 4 Zahlenwerten"
(Das ist die Überschrift des Programms)

Eingabe: 30 INPUT "Bitte Zahl 1 eingeben : ";A

Mit dieser Programmzeile haben wir die erste Variable eingelesen. Der Befehl INPUT dient zur Abfrage über den Bildschirm. Die Zeichen zwischen den Anführungszeichen werden wie eine PRINT-Anweisung behandelt. Sie können aber auch die Abfrage weglassen. Die Zeile würde dann folgendes Format haben:

```
30 INPUT A
```

Der Computer gibt bei der INPUT-Anweisung ein Fragezeichen aus. Die Abfrage oder der Hinweis dienen dem Benutzer des Programms zur Orientierung.

Für den ersten Zahlenwert haben wir den Buchstaben A als Variable gewählt. Es kann auch jeder andere Buchstabe verwendet werden. Auch können mehrere Buchstaben und Zahlen Verwendung finden. Das genaue Format in Applesoft-BASIC muß wie folgt aussehen:

- Das erste Zeichen muß ein Buchstabe sein

- Jedes weitere Zeichen kann ein Buchstabe oder ein Zahl sein

- Eine Variable kann bis zu 238 Buchstaben oder Zahlen enthalten

Bitte geben Sie noch die folgenden Zeilen ein:

```
40 INPUT "Bitte Zahl 2 eingeben : ";B
50 INPUT "Bitte Zahl 3 eingeben : ";C
60 INPUT "Bitte Zahl 4 eingeben : ";D
```

Diese dienen, wie Programmzeile 30, zur Abfrage der vier geforderten Zahlenwerte.

```
70 E = A+B+C+D
```

Zeile 70 definiert die Variable E. Der Computer hat das Programm im Arbeitsspeicher abgelegt. Wenn Sie das Programm mit dem LIST-Befehl anschauen, müßte es folgendermaßen aussehen:

```
10 HOME
20 PRINT"Addition von 4 Zahlenwerten"
30 INPUT"Bitte Zahl 1 eingeben : ";A
40 INPUT"Bitte Zahl 2 eingeben : ";B
50 INPUT"Bitte Zahl 3 eingeben : ";C
60 INPUT"Bitte Zahl 4 eingeben : ";D
70 E = A+B+C+D
```

Starten Sie das Programm mit RUN. Der Computer löscht den Bildschirm, schreibt "Addition von 4 Zahlenwerten" und fragt nach der ersten Zahl:

Bitte Zahl 1 eingeben :

Wir geben eine Zahl (z.B. 12) ein und schließen mit RETURN ab. Genauso verfahren wir mit den nächsten drei Zahlen (z.B. 13, 14, 15). Zeile 70 wird vom Computer ausgeführt, ohne daß Sie etwas auf dem Bildschirm sehen können. Sie sollten jetzt, nachdem das Programm beendet ist, die Variablen und deren Werte im Arbeitsspeicher abfragen.

Eingabe:

```
ÜPRINT A,B,C,D,E
```

Ausgabe z.B.:

```
12          13          14
15          54
```

Das Programm, das wir erstellt haben, braucht noch die Ausgabeanweisung des Ergebnisses. Dazu geben wir folgende Programmzeile ein:

```
80 PRINT "Das Ergebnis ist ";E
```

Jetzt ist unser kleines Programm fertig. Wir können es starten und sehen z.B. folgende Darstellung am Bildschirm:

```
Addition von 4 Zahlenwerten
Bitte Zahl 1 eingeben : 12
Bitte Zahl 2 eingeben : 13
Bitte Zahl 3 eingeben : 14
Bitte Zahl 4 eingeben : 15
Das Ergebnis ist 54
```

Numerische Variable werden Ihnen immer wieder bei der Programmierung begegnen. Häufig benötigte Zahlenwerte, sogenannte Konstante, können mit Hilfe von Variablen ersetzt werden. Dazu ein Beispiel:

Die Konstante Pi = 3.14159265.....

Sie wird durch die Variable PI ersetzt. Nun kann sie immer wieder im Programm verwendet werden, ohne daß die ganze Zahl neu eingegeben werden muß.

Durch einfaches Einsetzen eines Buchstabens können wir eine Variable definieren. Der Rechner definiert selbst, um welche Art einer Variablen es sich handelt. Der Computer verwendet dazu die letzte Stelle des Variablennamens. Wenn der BASIC-Interpreter kein spezielles Zeichen vorfindet, so geht er davon aus, daß es sich um eine numerische Variable handelt. In Integer-BASIC können es nur ganze Zahlen sein. Folgende Variablennamen sind gültig:

| | |
|---|---|
| A | AA |
| B1 | OMEGA |
| C20 | DM |
| WERT1 | PF1 |

### 4.13.6. Textvariable

Die Anweisung für Textvariable muß immer mit einem Dollarzeichen ($) abgeschlossen werden. Es ist darauf zu achten, daß das erste Zeichen immer ein Buchstabe ist. Alle folgenden Zeichen können sowohl Buchstaben und Zahlen als auch Sonderzeichen sein. Als Beispiel folgende Variablennamen:

| | |
|---|---|
| GEWINN$ | BB$ |
| A1$ | ZEIT$ |
| N$ | BEZ$ |
| JAHR$ | KURS$ |

### 4.13.7. Ganzzahl-Variable

Ganzzahlige Variable haben an der letzten Position ein Prozentzeichen. Dezimalstellen werden auf- oder abgerundet. Es wird immer eine ganze Zahl ausgegeben.

Beispiel:

A% oder B1%

## 4.14 Der Variablenaufbau in INTEGER-BASIC

- Das erste Zeichen muß ein Buchstabe sein.
- Die Variable kann bis zu 100 Buchstaben oder Zahlen enthalten.
- Das zweite bis hundertste Zeichen können Buchstaben oder Zahlen sein.
- Textvariable werden am Ende mit einem Dollarzeichen gekennzeichnet.
- Numerische Variable haben keine besondere Kennzeichnung und liegen im Bereich -32767 bis 32767.

## 4.15 Der Variablenaufbau in APPLESOFT-BASIC

- Das erste Zeichen muß ein Buchstabe sein.
- Die Variablenbezeichnung kann bis zu 238 Zeichen lang sein.
- Das 2. - 238.Zeichen können Buchstaben oder Zahlen sein.
- Textvariable werden am Ende mit einem Dollarzeichen markiert.
- Ganze Zahlen sind am Schluß mit einem Prozentzeichen ausgewiesen und liegen im Bereich von -32767 bis 32767.
- Numerische Variable haben keine besondere Kennzeichnung und liegen zwischen $-10^{38}$ und $10^{38}$.

## 4.16 Reservierte Worte

Bei der Definition von Variablen muß darauf geachtet werden, daß sie nicht mit BASIC-Anweisungen verwechselt werden. Hätten wir z.B. die Variable AUTO gewählt, so ließe der BASIC-Interpreter das reservierte Wort TO heraus und würde es bei der Listingausgabe neu formatieren.

```
Eingabe:     10 AUTO$ = "Jahr"
             LIST

Ausgabe:     10 AU TO $ = "Jahr"
```

**LET**

Wird eine Variable festgelegt, so kann dieser Definition die Anweisung LET vorangestellt werden. Diese weist der Variablen einen Wert oder eine andere Variable zu. Aber auch schon das Gleichheitszeichen allein ersetzt LET vollständig.

Beispiel:

```
LET A = 100

oder

A = 100
```

Beide Schreibweisen sind gültig und haben die gleiche Bedeutung.

## 4.17 Operatoren

Wenn eine Addition ausgeführt wird, so sorgt der Operator für die richtige Abwicklung. Das Pluszeichen ist bei einer Addition der Operator. Er steuert den Rechenvorgang. Ein Pluszeichen wird als arithmetischer Operator bezeichnet. Auch die Zeichen - / * ^ sind arithmetische Operatoren. Sie helfen dem Computer, die richtigen Rechenschritte zu durchlaufen. Die Abwicklung der Anweisungen

verläuft nach bestimmten Regeln, die im BASIC-Interpreter gespeichert sind. Nachfolgend die Reihenfolge, in der die Rechenschritte bearbeitet werden:

| | | |
|---|---|---|
| 1. | ( ) | Klammern |
| 2. | - | negatives Vorzeichen |
| 3. | ^ | Potenz |
| 4. | * | Multiplikation |
| 4. | / | Division |
| 4. | MOD | Divisionsrest |
| 5. | + | Addition |
| 5. | - | Subtraktion |

Anweisungen, die in Klammern stehen, werden zuerst bearbeitet. Dabei sind die innersten Klammern den äußeren übergeordnet.

Beispiel:

```
PRINT (5-2)*2+(10+20)
36

PRINT ((20+5)*6+17)+4
171
```

Die Anweisungen werden entsprechend der obigen Aufstellung ausgeführt. Dabei haben Klammern die höchste Priorität. Anweisungen mit gleichen Operatoren werden von links nach rechts abgearbeitet. Auch Texte können mit Operatoren verknüpft werden. Integer-BASIC bietet diese Art der Textzusammenfassung nicht. In Applesoft-BASIC kann nur der Operator "+" zur Verknüpfung von Textteilen herangezogen werden.

Beispiel in Applesoft-BASIC:

| | | |
|---|---|---|
| "Hand"+"buch" | = | Handbuch |
| "Größe"+GR | = | Größe + dem Wert der Variablen GR |

| | | |
|---|---|---|
| AB$ + BB$ + CB$ | = | z.B.: AB$ = "Haus"<br>BB$ = "manns"<br>CB$ = "kost"<br>wird zu "Hausmannskost" |

## 4.18 Vergleichende Operatoren

Mit den vergleichenden Operatoren können Texte, Variable und Zahlen miteinander verglichen werden. Dabei gibt es zwei Zustände:

| | | |
|---|---|---|
| richtig | = | 1 |
| falsch | = | 0 |

Bei vergleichenden Operatoren gibt es keine Rangfolge. Sie werden immer von links nach rechts bearbeitet.

| | |
|---|---|
| < | kleiner als |
| > | größer als |
| = | gleich |
| >= | größer als oder gleich in Integer-BASIC |
| <= | kleiner oder gleich in Applesoft-BASIC |
| <> oder >< | ungleich in Applesoft-BASIC |
| # | ungleich in Integer-BASIC |
| >= oder => | größer als oder gleich in Applesoft-BASIC |
| <= oder =< | kleiner als oder gleich in Applesoft-BASIC |

In Integer-BASIC können nur gleich und ungleich für die Zuordnung von Texten verwendet werden. Alle anderen vergleichenden Anweisungen können nur numerische Daten beinhalten. Texte oder Zahlen müssen sowohl links als auch rechts vom Operator stehen.

## 4.19 Boolesche Algebra

Die Booleschen Anweisungen AND, OR, NOT haben alle die gleiche Rangfolge und werden von links nach rechts abgearbeitet. Dabei gilt:

| | | |
|---|---|---|
| 1 AND 1 = 1 | 1 OR 1 = 1 | NOT 1 = 0 |
| 1 AND 0 = 0 | 1 OR 0 = 1 | NOT 0 = 1 |
| 0 AND 1 = 0 | 0 OR 1 = 1 | |
| 0 AND 0 = 0 | 0 OR 0 = 0 | |

Booleschen Variablen wird immer der Wert 0 oder 1 zugeordnet. Die Verknüpfung mit Booleschen Operatoren bestimmt, welche Entscheidungen im Programm getroffen werden.

## 4.20 Tabellen

Große Datenmengen würden die Anwendung einer Variablenanweisung sprengen. Darum gibt es sogenannte Matrizen. Es wird also nicht mehr jedem Datensatz eine Variable zugeordnet, sondern nur noch ein Platzanweiser in einer Tabelle. Die Variable bekommt hierbei eine Nummer. Das ist mit einer Stadt zu vergleichen. Um die Übersicht zu behalten, werden nicht allen Häusern verschiedene Hausnummern zugeteilt, sondern es gibt Straßennamen in denen die Häuser durchnumeriert sind. Genauso verhält es sich mit unserer Matrize.

Beispiel:

Fünf verschiedene Werte sollen in einer Tabelle angeordnet werden. Würden wir verschiedene Variable benutzen, sähe das wie folgt aus:

AB CD HG KJ FD

Als Tabelle schaut dieses Modell wesentlich übersichtlicher aus:

A(1) A(2) A(3) A(4) A(5)

Diese Darstellungsweise bezeichnet man als Zeile einer Tabelle. Stehen die Daten untereinander, so spricht man von einer Spalte. Mit dieser Tabelle können die Daten übersichtlicher bearbeitet werden.

Es können aber auch Daten in Zeilen und Spalten verwaltet werden. Dann spricht man von mehrdimensionalen Tabellen. Zweidimensionale Tabellen werden in Spalten nebeneinader und Zeilen übereinander angeordnet. Beispiel:

Unsere Tabelle soll auf 4 Datensätze pro Spalte erhöht werden.

| | | | | |
|---|---|---|---|---|
| A(1,1) | A(1,2) | A(1,3) | A(1,4) | A(1,5) |
| A(2,1) | A(2,2) | A(2,3) | A(2,4) | A(2,5) |
| A(3,1) | A(3,2) | A(3,3) | A(3,4) | A(3,5) |
| A(4,1) | A(4,2) | A(4,3) | A(4,4) | A(4,5) |

Die einzelnen Tabellenzellen bezeichnet man auch als Elemente. Die Werte, die in Klammern stehen, werden als Indizes bezeichnet. Die erste Zahl gibt dabei die Zeile an. Der zweite Index bezeichnet die Spaltennummer.

1,1: 1 = 1.Zeile/1 = 1.Spalte
1,2: 1 = 1.Zeile/2 = 2.Spalte
2,1: 2 = 2.Zeile/1 = 1.Spalte usw.

In Integer-BASIC können nur eindimensionale Tabellen definiert werden. Applesoft-BASIC erlaubt die Erstellung von Tabellen bis zu 88 Dimensionen. Im Normalfall werden zweidimensionale Tabellen verwendet. Eine Tabelle kann nur eine Art von Zahlen enthalten. In Applesoft-BASIC können sowohl ganze Zahlen als auch Gleitkommazahlen Verwendung finden.

Der frei verfügbare Speicherbereich bestimmt die Anzahl der Indizes. Bevor mit einer Tabelle gearbeitet werden kann, muß die Größe bestimmt werden. Dies erfolgt mit der Anweisung

**DIM**

Integer-BASIC läßt nur eindimensionale Tabellen zu. Darum wird der Befehl DIM abweichend von Applesoft-BASIC formuliert.

**Für Integer-BASIC gilt:**

DIM vvv (iii) oder DIM vvv (iii),vvv (iii)

vvv = Variable
iii = Indizes

Beispiel in Integer-BASIC:

10 DIM A(12)

(Die Tabelle A kann mit maximal 13 Elementen besetzt werden)

10 DIM B(34)

(Es können 35 Elemente der eindimensionalen Tabelle B zugeordnet werden)

Wird der vordefinierte Wert überschritten, so gibt der Rechner folgende Fehlermeldung aus:

***** RANGE ERR**

Handelt es sich um eine Textvariable, so gibt der Computer die Fehlermeldung:

***** STRING ERR**

Numerischen Variablen werden keine Werte zugeordnet, sie müssen anschließend definiert werden. Textvariablen wird ein Scheintext von 0 Zeichen zugeordnet.

Applesoft-BASIC erlaubt die Definition von mehrdimensionalen Tabellen. Dabei müssen Tabellen mit weniger als 11 Elementen in jeder Dimension nicht vordefiniert werden. Der BASIC-Interpreter ordnet jeder Tabellenzelle den Höchstwert 10 zu.

**Für Applesoft-BASIC gilt:**

```
DIM vvv(iii)

DIM vvv(iii,iii)

DIM vvv(iii,iii),vvv(iii)

DIM vvv(iii,iii),vvv(iii,iii,iii,iii)
```

Es werden bis zu 88 Dimensionen verarbeitet. Wird diese Anzahl überschritten, so meldet der Computer:

**? OUT OF MEMORY ERROR**

Wird die dimensionierte Zahl überschritten, so erscheint die Fehlermeldung:

**?BAD SUBSCRIPT ERROR**

Die festgelegten Höchstwerte können im laufenden Programm nicht geändert werden. Wird es versucht, meldet der Rechner:

**?REDIM'D ARRAY ERROR**

Beispiele:

```
DIM V(17), M(34), H5(867)
```

(Der eindimensionalen Tabelle V werden 18 Elemente zugeordnet. In Tabelle M sind es 35, in Tabelle H5 sind es 868)

```
DIM S(12,12), GG(15,15,15)
```

Tabelle S ist zweidimensional. Sie kann 12 Zeilen und 12 Spalten umfassen. Tabelle GG ist dreidimensional und kann in jeder Dimension mit 15 Elementen belegt werden.

## 4.21 DATA und READ

Um größere Datenmengen verwalten zu können, muß die DATA-Anweisung benutzt werden. Mit ihr ist es möglich, rationell mehreren Variablen einen Wert zuzuordnen. Der DATA-Anweisung folgen die Daten, die zugeordnet werden. Sie sind durch Kommas getrennt. Zum Lesen dieser Daten benutzen wir die Anweisung

**READ**

Diese Anweisung folgt in einer weiteren Programmzeile und enthält z.B. die Variablennamen, denen die Werte in der DATA-Zeile zugeordnet werden.

Beispiel:

```
10 DATA 10,20,30,40
20 READ A,B,C,D
```

READ hat der numerischen Variablen A den Wert 10 zugeordnet. Der Variablen B wurde 20, C = 30 und D = 40 zugeordnet. Sind mehrere DATA-Zeilen nacheinander angeordnet, dann werden die DATA-Daten beim Auslesen aneinandergehängt.

Beispiel:

```
10 DATA 10,20,30,40,50
20 DATA 60,70,80,90,100
30 READ A
40 PRINT A
50 GOTO 30
```

Wird dieses Beispiel mit RUN gestartet, ordnet die Anweisung READ der Variablen A so lange Werte zu, bis 100 erreicht wird. Da keine weiteren Werte mehr vorhanden sind, erscheint die Fehlermeldung:

**? OUT OF DATA ERROR IN 30**

Mit einer IF..THEN-Anweisung kann A abgefragt werden. Ist A grösser als 100, dann verzweigt es weiter im Programm. Diese Möglichkeit bespreche ich im nächsten Abschnitt genauer. Schauen wir uns zunächst folgendes Beispiel an:

```
10 DATA 10,20,30,40
20 READ A,B,C,D
30 RESTORE
40 READ E,F,G,H
50 PRINT A,B,C,D,E,F,G,H
RUN
```

```
10          20          30
40          10          20
30          40
```

Die Programmzeile 20 definiert die Werte A = 10, B = 20, C = 30 und D = 40. In Zeile 30 steht der neue BASIC-Befehl RESTORE. Er bewirkt, daß die neue READ-Anweisung in Zeile 40 beim DATA-Wert 10 von neuem beginnt. Den Variablen E, F, G und H werden wieder die Werte 10, 20, 30 und 40 zugeordnet. Die Anweisung RESTORE wird insbesondere dann verwendet, wenn in längeren Programmen öfter auf den gleiche DATA-Bestand zugegriffen wird.

**CLEAR**

Mit dieser Anweisung werden die Werte aller definierten Variablen auf 0 gesetzt. Texte in Textvariablen werden gelöscht. Diese Anweisung kann nur in Applesoft-BASIC verwendet werden.

Für Integer-BASIC kann die Anweisung

**CLR**

benutzt werden. Sie ist nur in der Betriebsart Direktanweisung einsetzbar. Dimensionierte Variable werden durch CLR aufgehoben.

## 4.22 Programmaufbau und Ablauf

Der größe Vorteil von BASIC-Programmen ist die Möglichkeit, den Programmablauf nach jeder Eingabe zu überprüfen. Eine Anweisung wird eingegeben, und sofort kann das Resultat begutachtet werden. Dabei leisten die Zeilennummern einen großen Dienst. Sie helfen den Überblick zu behalten. Auch der Computer kann ohne diese Zeilennumerierung nicht auskommen. Sie dienen ihm als Wegweiser. So ist es möglich, das Programm zu verschachteln und dadurch den Programmieraufwand zu senken. Im folgenden Abschnitt sollen die verschiedenen Programmsteuerungen näher betrachtet werden.

## 4.23 Sprunganweisung

Verschiedene Programmteile werden in BASIC-Programmen öfters aufgerufen. Damit Sie die Programmteile nicht immer wieder eintippen müssen, gibt es die Sprunganweisungen.

**GOTO**

weist den Computer an, eine bestimmte Programmzeile als nächste abzuarbeiten. Dabei wird die erste Anweisung der angesprungenen Zeile zuerst bearbeitet.

Beispiel:

```
Eingabe: 10 PRINT "Test"
         20 PRINT "Tag"
         30 GOTO 60
         40 PRINT "Monat"
         50 PRINT "Jahr"
         60 PRINT "Hallo"
         70 END

Ausgabe: Test
         Tag
         Hallo
```

Die Anweisung in Zeile 30 hat eine Verzweigung zu Zeile 60 eingeleitet. Die Programmzeilen 40 und 50 wurden übersprungen und kamen nicht zur Ausführung.

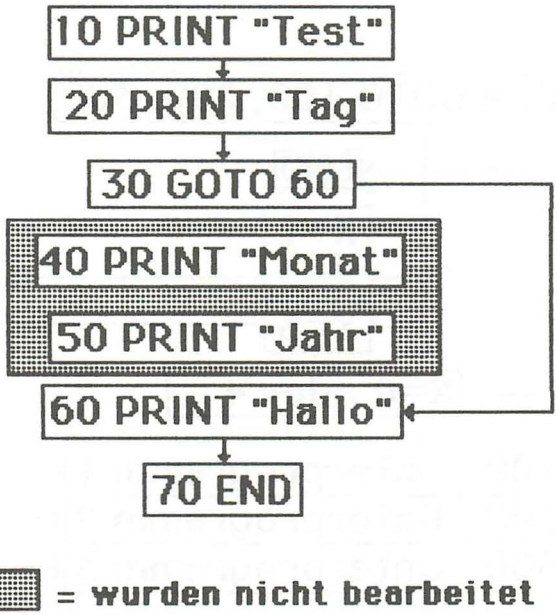



Abb. 4-8: GOTO-Anweisung

Es gibt die Möglichkeit, die GOTO-Verzweigung mit einer Bedingung zu verknüpfen. Dabei wird ihr eine Variable zugeordnet. Wenn der Wert der Variablen richtig gesetzt ist, verzweigt das Programm an die Stelle, die der GOTO-Anweisung folgt. Dazu verwenden wir die Anweisungsstruktur

**ON .. GOTO**

An einem Beispiel können Sie das am besten verstehen.

```
10 ON X GOTO 100,200,300
```

Die Variable X muß einen Wert zwischen 1 und 3 besitzen, damit eine Verzweigung erfolgen kann. Hat X den Wert 1, dann verzweigt das Programm zur Programmzeile 100. Wenn X den Wert 2 besitzt, so verzweigt das Programm zur Programmzeile 200 usw.

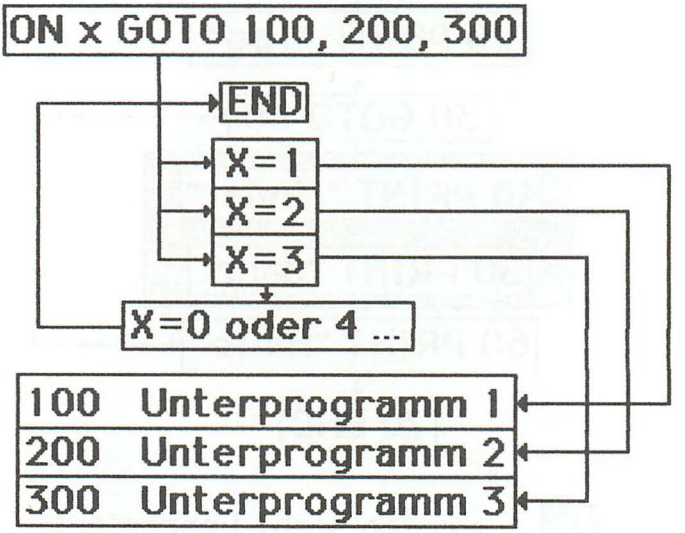



Abb. 4-9: ON-GOTO-Anweisung

```
10 GOTO 10*A+100
```

Bei diesem Beispiel wird die GOTO-Anweisung durch eine Berechnung eingeleitet. Die angesprungene Programmzeile resultiert aus dem Wert der Variablen X. So können Unterprogramme und andere Programmteile angesprungen werden.

## 4.24 Programmschleifen

Das Wiederholen von Anweisungen wird in BASIC-Programmen oft angewendet. Dabei durchläuft das Programm immer wieder einen bestimmten Programmteil. Ein kleines Programm kann auch nur aus einer Programmschleife bestehen. Um zum Beispiel die ASCII-Codes der Tastenbelegung zu ermitteln, muß ein bestimmter Programmteil diesen Code abfragen und am Bildschirm darstellen.

```
10 FOR I = 32 TO 127
20 PRINT CHR$(I); " = ";I
30 NEXT I
40 PRINT "Ende"
```

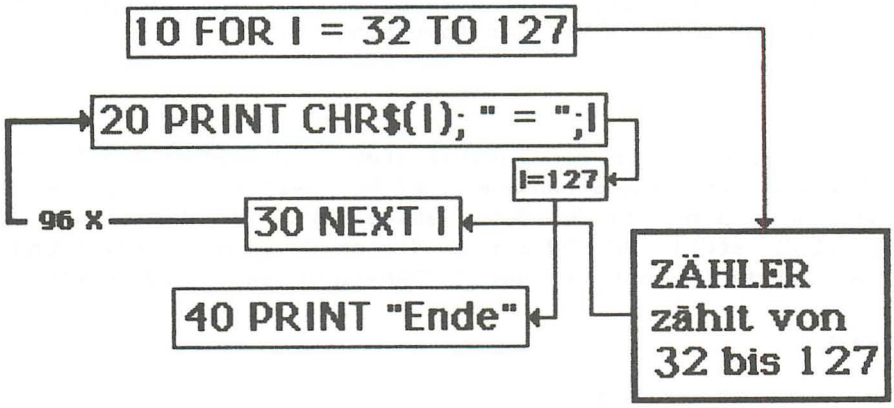



Abb. 4-10: Programmschleife

Der Computer zählt in der Zeile 10 von 32 bis 127. Dabei arbeitet er bei jedem Schritt alle Anweisungen durch, die zwischen FOR und NEXT stehen. Die Anweisung NEXT läßt das Programm wieder in Zeile 10 weiterzählen. Dabei wird der Wert von I immer um 1 erhöht. Ist der Wert 127 erreicht, verzweigt das Programm nicht mehr und geht zu Zeile 40.

Soll in Zweierschritten ausgegeben werden, so müssen Sie die Zeile 10 um die Anweisung

**STEP 2**

erweitern. Wird STEP nicht angegeben, so wird immer um den Wert 1 erhöht. Es können auch negative Zahlenwerte angegeben werden.

## 4.25 Das Unterprogramm

Wer öfters programmiert wird bald feststellen, daß viele Programmteile immer wieder vorkommen. Diesen Zustand können wir ausnutzen, um unsere Programme zu verkürzen. Ein Unterprogramm hilft uns dabei. Die Subroutine, wie man das Unterprogramm auch nennt, besteht aus einer Reihe von Anweisungen, die vom Hauptprogramm öfters benötigt werden. Mit der Anweisung

**GOSUB**

springt das Programm in ein Unterprogramm, das ab der Zeile ausgeführt wird, deren Zeilennummer dem GOSUB-Befehl folgt. Das Unterprogramm wird bis zu der Zeile ausgeführt, in der es auf den Befehl RETURN stößt. RETURN ist in diesem Fall eine Anweisung im Programm und darf nicht mit der RETURN-Taste verwechselt werden.

Beispiel:

```
10 GOSUB 100
15 PRINT A
20 GOSUB 100
50 PRINT A + 1000
100 A = 1000
150 A = A - 1
160 RETURN
```

Das Programm verzweigt bei Zeile 10 zum Unterprogramm in Zeile 100. Dieses wird benötigt, um die Anweisung in Zeile 15 durchführen zu können. In Zeile 20 wird das Unterprogramm erneut aufgerufen. RETURN bewirkt den Rücksprung auf Zeile 50.

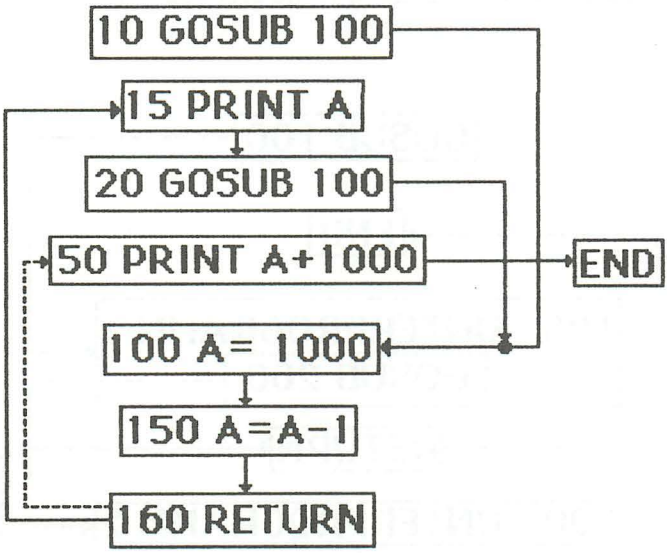



Abb. 4-11: Einfache GOSUB-Anweisung

Mehrere Unterprogramme werden auch häufig ineinandergeschachtelt. Dabei verzweigt das Programm so lange nach unten, bis es auf die Anweisung RETURN trifft. Dann wird das Unterprogramm ausgeführt, von dem zuletzt verzweigt wurde. Schauen wir uns das abgeänderte GOSUB-Programm an.

```
10 GOSUB 100
15 PRINT A
50 PRINT A+1000
100 A= 1000
105 GOSUB 200
110 A= A-1
120 RETURN
200 PRINT A
210 RETURN
```

Die Zeile 105 läßt das Unterprogramm in ein weiteres Unterprogramm verzweigen. Nach der Ausführung der Anweisung PRINT A in Zeile 200 springt das Programm zurück in Zeile 110, um die vorhergegangene Subroutine abzuarbeiten. Danach verzweigt auch das erste Unterprogramm wieder in das Hauptprogramm.

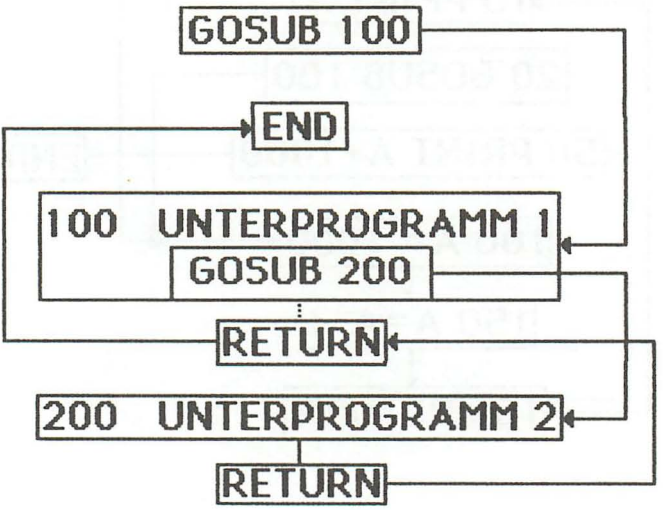



Abb. 4-12: Ineinandergeschachtelte Unterprogramme

Es ist dabei zu beachten, daß eine Unterroutine nie in sich selbst verzweigen kann. Auch kann eine zweite aufgerufene Unterroutine nicht in die erste verzweigen. Eine Ausnahme macht dabei die Rückkehr mit der Anweisung RETURN.

Wenn von einer Unterroutine wieder in das Hauptprogramm verzweigt werden soll, dieses Mal aber ohne RETURN zurückgesprungen wird, so kann das mit der Anweisung

**POP**

erfolgen. POP bewirkt, daß die Rücksprungadresse gelöscht wird. Mit einer GOTO-Anweisung kann die neue Rücksprungadresse angegeben werden. Beachten Sie, daß die Anweisung POP immer vor dem GOTO-Befehl kommen muß.

Wie schon bei der GOTO-Anweisung, so kann auch bei GOSUB eine bedingte Verzweigung erfolgen. Dies geschieht mit der Anweisung

**ON GOSUB**

Diese Anweisung wird in Integer-BASIC und Applesoft-BASIC unterschiedlich angewendet. Wie schon bei der ON GOTO-Anweisung, so wird auch hier eine Variable abgefragt. So können viele Unterroutinen mit Hilfe einer Variablen angesprungen werden.

Beispiel in Integer-BASIC:

```
100 GOSUB X*300/100
```

Aus der Errechnung der Zahlenwerte ergibt sich die Verzweigungsadresse. Wenn X = 2 ist, dann verzweigt das Programm zur Programmzeile 6. Man hätte natürlich auch X*3 schreiben können. Sollte die errechnete Zeilennummer nicht vorhanden sein, so meldet der Computer:

***** BAD BRANCH ERR**

In Applesoft-BASIC wird diese Anweisung anders formuliert:

```
100 ON X GOTO 100,200,300
```

Wie schon bei der ON-GOTO-Anweisung, kann auch hier die Variable die Ansprungzeile bestimmen. Hat X den Wert 1, so springt das Hauptprogramm in die BASIC-Zeile 100. Bei 2 springt es in die Zeile 200 usw. Wird in X eine Zahl außerhalb der vorhandenen Einsprungzeilen definiert, geht das Programm zur folgenden Zeile über.

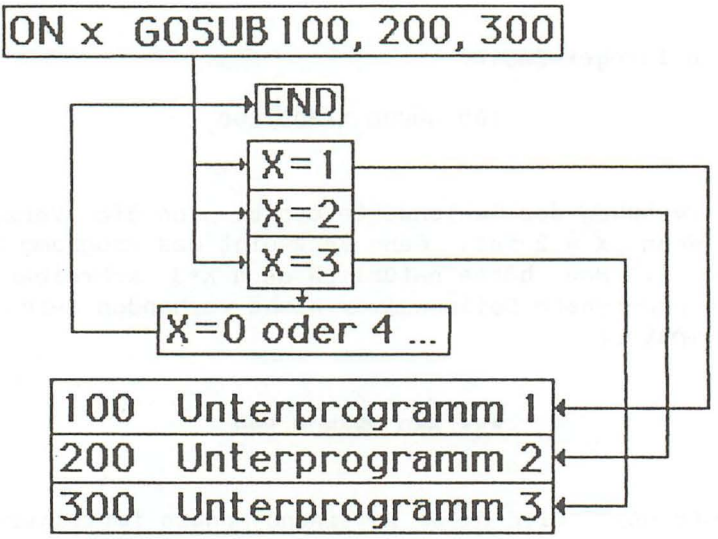



Abb. 4-13: ON-GOSUB-Anweisung

## 4.26 IF-THEN-Anweisung

Mit der Anweisung IF THEN ist es möglich, das Programm Entscheidungen fällen zu lassen. Nur wenn eine Bedingung erfüllt ist, kann die nächste Anweisung ausgeführt werden.

Applesoft kann bei einer bedingten GOTO-Anweisung auch ohne THEN auskommen. Folgendes Beispiel zeigt dies:

```
10 IF A = 10 THEN GOTO 100
```

oder

```
10 IF A = 10 GOTO 100
```

Werden mehrere Anweisungen nach einer IF-THEN-Bedingung in eine Programmzeile geschrieben, so wird in Applesoft die Anweisung nur dann ausgeführt, wenn auch die Bedingung erfüllt ist.

Beispiel:

```
10 IF A = 20  THEN B = 100:PRINT "Test"
```

In Applesoft wird die PRINT-Anweisung nur dann ausgeführt, wenn A = 20 ist. Integer-BASIC gibt das Wort "Test" auch dann aus, wenn A nicht 20 ist. In diesem Fall wird der PRINT-Befehl in die nächste Zeile geschrieben.

## 4.27 Ein- und Ausgabefunktionen

Bei der Abarbeitung von Programmen ist es nötig, mit dem Computer Kontakt aufzunehmen. Dazu zählt die Eingabe über die Tastatur genauso wie die Meldungen des Computers am Bildschirm.

Damit der Computer Meldungen, Ergebnisse und Werte ausgeben kann, benutzen wir die Anweisung PRINT. Schon am Anfang dieses Kapitels ist die Funktionsweise der Anweisung PRINT erläutert worden. Im folgenden Abschnitt lernen Sie einige weitere Möglichkeiten des Befehls kennen.

Die einfachste Art, einen Text auszugeben, erfolgt mit PRINT, gefolgt von dem Text in Anführungszeichen.

```
PRINT "Bitte geben Sie eine Zahl ein"
```

Bei der Abarbeitung wird nur das in Anführungszeichen stehende ausgegeben. Die Anführungszeichen gelten als Markierungen für den

BASIC-Interpreter. Um auch ein Anführungszeichen in den Text einfügen zu können, müssen wir einen kleinen Trick anwenden.

```
PRINT CHR$(34);" = Anführungszeichen"
```

Ergebnis:

```
" = Anführungszeichen
```

Durch das Abfragen des ASCII-Code für Anführungszeichen wurde die Markierung ausgegeben. So ist es möglich, auch innerhalb einer PRINT-Anweisung ein Anführungszeichen zu setzen. Wird das Anführungszeichen vergessen, so nimmt der BASIC-Interpreter an, daß es sich um eine Variable handelt. Wurde diese nicht zufällig benutzt, so ergibt die Abarbeitung dieser PRINT-Anweisung den Wert Null. Somit kann auch eine Variable ausgegeben werden. Folgen Zahlen und Operatoren dem PRINT-Befehl, so werden diese als mathematische Zeichen behandelt. Folgendes Beispiel soll dies erklären:

```
PRINT Text    (Text wurde nicht in Anführungszeichen gesetzt)
0

PRINT 2 + 4
6
```

Satzzeichen werden dazu benutzt, das Ausgabeformat der PRINT-Anweisung zu beeinflussen. Im Normalfall folgt jeder PRINT-Anweisung ein Zeilensprung an den Anfang der nächsten Bildschirmzeile. Zwei Satzzeichen verändern diesen Vorgang:

### 4.27.1 Das Komma in einer PRINT-Anweisung

Wie eine Schreibmaschine ist auch der Bildschirm des Computers mit unsichtbaren Tabulatoren versehen. Das sind Zeiger, die in gewissen Abständen den Bildschirm unterteilen. Mit dem Komma nach einer Ausgabeanweisung mit PRINT wird der nächste PRINT-Befehl an der Stelle ausgeführt, an der ein Tabulator steht. In INTEGER-BASIC und APPLESOFT-BASIC werden die Tabulatoren unterschiedlich gesetzt.

```
Applesoft:      ÜPRINT1,1,1,1,1,1
                1               1               1
                1               1               1

Integer:        >PRINT1,1,1,1,1,1
                1       1       1       1       1
                1
```

In Integer-BASIC wird die 1., 9., 17., 25. und 33. Spalte des Bildschirms verwendet. Applesoft-BASIC benutzt die 1., 17. und 33. Spalte. Das Komma ermöglicht es, drei- oder fünfspaltige Tabellen auf dem Bildschirm darzustellen. Texte und Variable können auf diese Weise überschaubar angeordnet werden. Ist eine Variable länger als der Abstand zwischen zwei Tabulatoren, so wird der nächste Tabulator zur Ausgabe benutzt.

```
PRINT"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA","BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBB"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

### 4.27.2 Das Semikolon in einer PRINT-Anweisung

Das Zusammensetzen von mehreren PRINT-Anweisungen kann mit dem Semikolon erfolgen. Dabei wird es zwischen die Ausgabeanweisungen gesetzt. Der Zeilensprung wird unterdrückt, und es erfolgt kein Sprung zum Zeilenanfang der nächsten Bildzeile. Erst nach Ablauf des letzten PRINT-Befehls wird auf die nächste Zeile geschaltet. Das Semikolon kann die Ausgabe der folgenden Programmzeilen beeinflussen. Wird die Programmzeile mit einem Semikolon abgeschlossen, so folgt die Ausgabe direkt danach.

```
PRINT1,1,1,1,1,1
111111

10 PRINT "Haus";
20 PRINT "tür"
RUN
Haustür
```

### 4.27.3 Tabulatoren setzen

Für die Darstellung von Texten, Tabellen und Hinweisen sind Tabulatoren sehr nützlich. Sie sollen verhindern, daß zum Beispiel für eine zentrierte Überschrift Leerzeichen eingefügt werden müssen. Der TAB-Befehl wird in Integer- und Applesoft-BASIC unterschiedlich formuliert. Als Beispiel soll eine Tabelle mit Namen auf dem Bildschirm dargestellt werden. Folgende Programmierschritte sind möglich:

In Applesoft-BASIC:

```
PRINT"Franz      Ingrid      Willy      Babsi"

PRINT"Franz";TAB(12);"Ingrid";TAB(24);"Wi
lly";TAB(35);"Babsi"

PRINT"Franz";:HTAB 12:?"Ingrid";:HTAB 24:
?"Willy";:HTAB 35:?"Babsi"
```

In Integer-BASIC:

```
PRINT"FRANZ      INGRID      WILLY      BABSI"

10 PRINT"FRANZ";
20 TAB 12
30 PRINT"INGRID";
40 TAB 24
50 PRINT"WILLY";
60 TAB 35
70 PRINT"BABSI"
80 END
```

Die Beispiele zeigen alle das gleiche Ergebnis. Im ersten Augenblick sieht es so aus, als wenn die Tabulatorbefehle keine Programmiererleichterung darstellten. Wenn Sie jedoch bedenken, daß die Zahlenwerte durch Variable ersetzt werden, so können umfangreiche Darstellungen mit nur wenigen Befehlen ausgeführt werden. Bei der Darstellung kleiner Zeichenmengen am Ende einer Bildschirmzeile zeigen diese Befehlsfolgen ihre große Nützlichkeit. Außerdem entfällt so das mühsame Zählen der Leerschritte. Mit dem TAB- oder HTAB-Befehl ist das kein Problem. Es wird nur die Zahl verändert, und schon steht Ihre PRINT-Ausgabe an einer anderen Stelle.

### 4.27.4 TAB

Diese Anweisung wird in Integer- und Applesoft-BASIC unterschiedlich behandelt. TAB darf in Applesoft-BASIC nur innerhalb einer PRINT-Anweisung benutzt werden. Alle anderen Versuche werden mit einem Syntax-Error quittiert. Im vorangegangenen Beispiel sind die Tabulatorstellen angegeben worden, wo die nächste Ausgabe erfolgen soll. Dabei muß die Tabulatorstelle in Klammern gesetzt werden.

In Integer-BASIC ist TAB ein abgeschlossener BASIC-Befehl. Die PRINT-Anweisung, die der Zeile mit dem TAB-Befehl folgt, wird an der Stelle ausgegeben, die bei der Anweisung TAB als ganze Zahl folgt.

### 4.27.5 HTAB

Die Anweisung HTAB kann nur in Applesoft benutzt werden. Sie ermöglicht eine horizontale Steuerung des Cursors. Eine PRINT-Anweisung wird an der Stelle fortgesetzt, die im HTAB-Befehl als ganze Zahl angegeben ist. Wie Sie im vorangegangenen Beispiel gesehen haben, benutzt HTAB nicht die festen Tabulatoren. Mit dieser Anweisung können Sie jede Stelle innerhalb einer Bildschirmzeile anwählen.

### 4.27.6 VTAB

In Applesoft-BASIC gibt es die Möglichkeit, auch vertikale Tabulatoren zu setzen. In diesem Fall sprechen wir von der vertikalen Cursorsteuerung auf dem Bildschirm. Durch die Anweisung **VTAB** wird der Cursor angewiesen, auf die Zeile zu springen, die dem VTAB-Befehl folgt. Diese Anweisung ist sehr nützlich in Verbindung mit dem Befehl HTAB.

### 4.27.7 SPC

In Applesoft-BASIC sind Leerschritte auch mit der Anweisung SPC einfügbar. Dabei muß hinter dieser Befehlsfolge in Klammern die Zahl der Leerschritte angegeben werden, die eingerückt werden sollen. Die Anweisung ist ein Bestandteil eines PRINT-Befehls und muß daher durch ein Semikolon von diesem getrennt werden. Wird ein Komma dazwischengestellt, ergibt sich die Ausgabe an der Tabulatorstelle, die SPC folgt. In Abbildung 4-14 wird die Darstellung, nach Abarbeiten der folgenden Befehlsfolge, auf dem Bildschirm schematisch dargestellt.

```
PRINT SPC(15);"Umsatzzahlen"
```

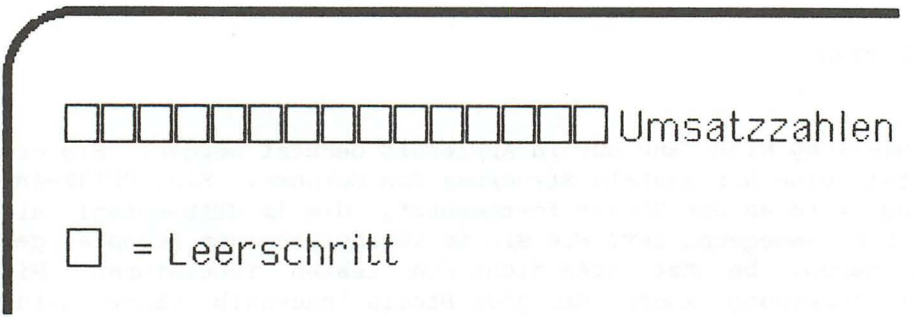



Abb. 4-14: PRINT-Anweisung mit SPC(15)

### 4.27.8 POS

Mit dieser Anweisung wird die Cursorposition festgestellt. Sie kann nur in Applesoft angewendet werden. Dabei zählt POS von 0 bis 39. In Klammern folgt immer eine Null, damit wird POS ein Pseudowert zugewiesen.

Beispiel:

```
10 PRINT POS(0)
```

Wird diese Anweisung ausgeführt, so gibt der Computer die Cursorposition in horizontaler Richtung an, die bei der Abarbeitung der Anweisung POS vorlag. In der Betriebsart Einzelanweisung kann dieser Befehl ebenfalls angewendet werden.

## 4.28 Darstellungsvarianten

In Applesoft-BASIC ist es möglich, Zeichen am Bildschirm invers darzustellen. Dazu ist vor die PRINT-Anweisung der Befehl

**INVERSE**

zu setzen. Er wird durch einen Doppelpunkt von PRINT getrennt. In Bild 4-5 wird die unterschiedliche Darstellungsweise gezeigt. Geben Sie bitte folgende Befehlsfolge ein:

```
NEW (löscht den BASIC-Arbeitsspeicher!)

10 PRINT "DIESER TEXT IST NORMAL DARGESTELLT"

20 INVERSE

30 PRINT "DIESER TEXT IST INVERS DARGESTELLT"
```

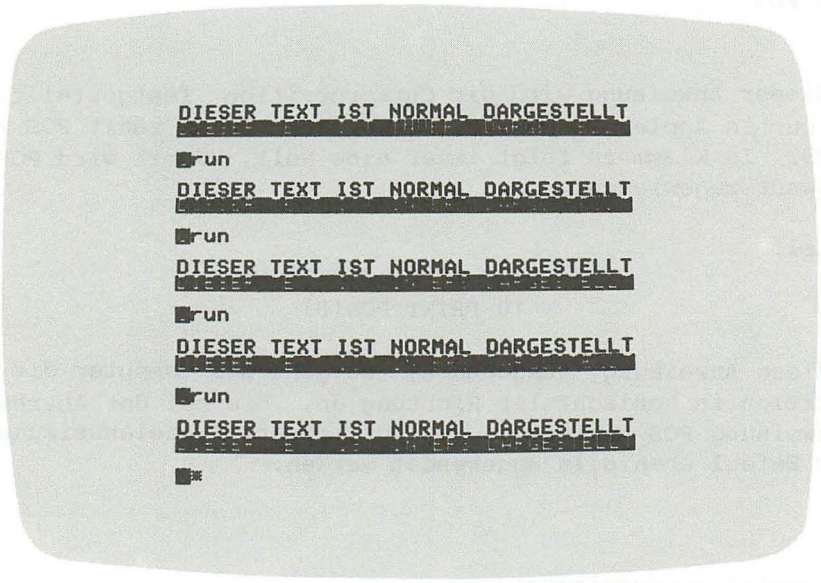



Abb. 4-15: Inverse Darstellung

Die inverse Ausgabe bleibt so lange erhalten, bis eine andere Anweisung diesen Modus zurücksetzt. Das geschieht mit der Anweisung

**NORMAL**

Die Zeichen, die ausgegeben werden sollen, dürfen nur GROSSBUCHSTABEN sein. Kleinbuchstaben würden zu einer Ausgabe von Zahlen und Sonderzeichen führen.

Mit der Anweisung **FLASH** werden alle anschließend ausgegebenen Zeichen blinkend dargestellt. Auch dieser Modus kann mit **NORMAL** wieder zurückgesetzt werden. Sie können mit POKE 50,127 den gleichen Darstellungseffekt erzielen. Den Kleinbuchstaben ist dagegen kein Charactercode für FLASH zugewiesen.

## 4.29 Dialogprogrammierung

In vielen Programmen ist es nötig, bestimmte Daten in das Programm einfließen zu lassen. Dazu stellt uns BASIC einige gute Programmiermöglichkeiten zu Verfügung.

## 4.30 Eingabefunktion GET

Sie haben z.B. ein Auswahlmenü programmiert und wollen durch einzelne Buchstaben die Programmteile aufrufen. Um diese Programmiersituation in den Griff zu bekommen, benutzen Sie den Befehl **GET**. Bearbeitet der BASIC-Interpreter die Anweisung, verlangt er nur ein einziges Zeichen. Die GET-Anweisung muß mit einer Variablendefinition abschließen. Trifft der BASIC-Interpreter auf diese Anweisung, wartet er so lange, bis ein Zeichen eingegeben wird. Dabei unterscheidet er zwischen numerischen und Stringvariablen.

Beispiel:

```
10 GET A
20 PRINT "A = ";A
```

| Eingabe: | Ergebnis: |
|---|---|
| RUN | |
| 1 | a = 1 |
| RUN | ?SYNTAX ERROR |
| A | |

Die Eingabe der Zahl 1 ergab das Ergebnis a=1. Der Buchstabe A war keine numerische Variable. Darum wurde eine Fehlermeldung ausgegeben. Bei diesem Beispiel ist es auch nicht möglich, zwei oder mehr Zahlen einzugeben. Denn die Anweisung GET nimmt nur ein Zeichen entgegen und geht dann zur nächsten Programmzeile. Es gibt jedoch die Möglichkeit, auch Buchstaben oder Sonderzeichen einzugeben. Dazu folgendes Beispiel:

```
10 GET A$
20 PRINT "A$ = ";A$
```

Es wurde eine Zeichen- oder Stringvariable definiert. Sie können jetzt auch Buchstaben und Sonderzeichen eingeben. In einer bedingten IF..THEN-Anweisung kann auch ein ganz bestimmtes Zeichen abgefragt werden.

Beispiel:

```
10 PRINT "A = Unterprogramm 1"
15 GET U$
40 IF U$ <>"A" THEN GOTO 15
100 PRINT:PRINT:PRINT"Unterprogramm 1"
```

Nach dem Starten des Programms wartet der Computer so lange mit der Abarbeitung des "Unterprogramm 1", bis A eingegeben wurde. Der Kleinbuchstabe a wird nicht akzeptiert.

## 4.31 Eingabefunktion INPUT

Sollen in unserem Beispiel mehrere Programmteile aufrufbar sein, so müssen wir den Befehl INPUT einsetzen. Mit ihm ist es auch möglich, längere Worte und ganze Sätze abzufragen.

Beispiel:

```
10 PRINT "A = Unterprogramm 1"
20 PRINT "B = Unterprogramm 2"
30 PRINT "C = Unterprogramm 3"
40 INPUT X$
50 IF X$ = "A" THEN GOTO 100
60 IF X$ = "B" THEN GOTO 200
70 IF X$ = "C" THEN GOTO 300
80 GOTO 40
100 PRINT "UNTERPROGRAMM 1":END
200 PRINT "UNTERPROGRAMM 2":END
300 PRINT "UNTERPROGRAMM 3":END
```

Anhand dieses Bespiels können Sie ein kleines Dialogprogramm aufbauen. Durch die INPUT-Funktion wird eine Variable definiert, die im folgenden Programmablauf von Bedeutung ist. So werden numerische und Zeichenvariable in ein Programm eingelesen. Durch Komma getrennt, können auch mehrere Variable einer INPUT-Funktion folgen. Bei der Abfrage einer INPUT-Funktion wird ein Fragezeichen ausgegeben.

Beispiel in Applesoft:

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 INPUT A$ | ? TEXT |
| 20 PRINT A$ | TEXT |
| RUN | |
| TEXT | |

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 INPUT A$,B$ | ? TEXT |
| 20 PRINT A$;B$ | ?? BAUSTEIN |
| RUN | TEXTBAUSTEIN |
| TEXT | |
| BAUSTEIN | |

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 INPUT A | ? TEXT |
| 20 PRINT A | ? REENTER |
| RUN | ? 122 |
| TEXT | 122 |
| 122 | |

Die Beispiele zeigen, daß die Anweisung INPUT ein sehr nützliches Programmierwerkzeug darstellt. Wird die Eingabe nicht näher definiert, so gibt der Computer **REENTER** aus. Wollen Sie Ihr Programm auch anderen Computerbenutzern zugänglich machen, müssen Sie bei einer Abfrage Hinweise einfügen. Dies braucht nicht durch einen PRINT-Aufruf zu geschehen. In die INPUT-Anweisung kann bereits ein Hinweis eingefügt werden.

Beispiel:

```
10 INPUT "Geben Sie eine Zahl zwischen 0 und 100 ein !", A
20 PRINT "Eine Zahl zwischen 1 und 100 ist ";A
30 END
```

Im Gegensatz zu GET muß die Eingabe bei INPUT mit RETURN abgeschlossen werden. Wird ein Hinweis benutzt, muß hinter der Mitteilung bei Applesoft ein Semikolon folgen. In Integer-BASIC wird ein Komma gesetzt. Außerdem sind in Integer-BASIC einige Hinweise zu beachten. Folgende Beispiele sollen das anschaulich erläutern.

Beispiel:

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 INPUT A | ?123 |
| 20 PRINT A | 123 |
| 30 END | |
| RUN | |
| 123 | |
| | |
| RUN | ?TEXT |
| TEXT | *** SYNTAX ERR |
| | RETYPE LINE |
| | ? |

Integer-BASIC hat den TEXT abgelehnt. Es handelt sich um keine Textvariable. Kann der BASIC-Interpreter das Eingegebene nicht verstehen, gibt er in Integer-BASIC ***** SYNTAX ERR RETYPE LINE** aus. Sie müssen die Zeile nochmals eingeben.

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 INPUT A$ | 123 |
| 20 PRINT A$ | *** STR OVFL ERR |
| 30 END | STOPPED AT 10 |
| RUN | > |
| 123 | |
| | |
| RUN | A |
| A | A |

Wird eine Textvariable definiert, so werden auch Zahlen nicht als Zahlen, sondern als Zeichen behandelt. Es ist dabei darauf zu achten, daß ein Zeichen nicht überschritten wird. Denn der Integer-BASIC-Interpreter läßt keine längeren Eingaben zu, wenn diese nicht in einer DIM-Anweisung angegeben werden. Wird der INPUT-Variablen eine Länge mit DIM zugeordnet, so kann diese kleiner, jedoch nicht größer eingegeben werden. Es würde die Fehlermeldung ***** STR OVFL ERR** erscheinen.

Beispiel:

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 DIM A$ (10) | TEXT |
| 20 INPUT A$ | TEXT |
| 30 PRINT A$ | > |
| 40 END | |
| RUN | |
| TEXT | |

## 4.32 Warteschleifen

Besonders bei Überprüfung des Programmablaufs ist es von Bedeutung, eine Programmierhilfe zu besitzen, die es uns ermöglicht, Programme anzuhalten und wieder fortzusetzen. Bei der Programmierung in BASIC werden Sie schon bald feststellen, daß Ihre Programme eine Länge bekommen, die oft nicht mehr überschaubar ist. Um dennoch den Programmablauf verfolgen zu können, muß eine Bremse an schwierigen Stelle eingebaut sein, um die Bildschirmausgabe besser betrachten zu können. Dazu benutzen wir den Befehl

**STOP**

Trifft der BASIC-Interpreter auf diese Anweisung, hält das Programm an. Die gleiche Situation haben wir beim Drücken der Tasten CONTROL und C oder bei der Anweisung END. Nach END kann das Programm nur dann fortgesetzt werden, wenn END nicht am Ende des Programms steht, sondern nur einen Programmteil beendet. Es folgt die Meldung

**BREAK IN __**

Sie zeigt an, in welcher Programmzeile das Programm gestoppt wurde. Um es wieder fortzusetzen, dürfen Sie nicht RUN eingeben. Die Folge wäre, daß das Programm wieder am Anfang beginnen würde. Auch dafür hat der BASIC-Interpreter eine Anweisung. Sie heißt:

**CONT**

Das ist die Abkürzung vom englischen Wort continue, was soviel bedeutet wie fortsetzen. Wird dieser Befehl eingegeben und mit RETURN bestätigt, fährt das Programm bei der Programmzeile fort, die dem CONT-Befehl folgt. In Integer-BASIC wird das Fortsetzen des Programms mit **CON** ermöglicht. Es kann aber nur dann fortgesetzt werden, wenn inzwischen keine Änderungen vorgenommen wurden. Wird das Programm innerhalb einer INPUT-Anweisung angehalten, kann auch mit CONT oder CON nicht mehr gestartet werden.

**SPEED**

Zur Verlangsamung einer Bildschirmdarstellung oder zur Zeitlupenstudie einer Grafikdarstellung stellt uns die Anweisung **SPEED** die Möglichkeit zur Verfügung, die Ausgabegeschwindigkeit zwischen zwei Zeichen zu verlangsamen.

Beispiel:

Sie wollen die Ausgabe eines Textes, den Sie in eine PRINT-Anweisung gefaßt haben, in Lesegeschwindigkeit am Bildschirm ausgeben.

Sie geben vor der PRINT-Anweisung folgenden Befehl ein:

SPEED = 50

Um wieder zur Normalgeschwindigkeit zurückzukehren, geben Sie nach der PRINT-Anweisung SPEED=255 ein. Auch die Übertragung zu Peripheriegeräten wird mit diesem Befehl verlagsamt. Die Skala von SPEED reicht von 0 bis 255.

**WAIT**

Ist ein kurzzeitiges Anhalten des Programms erwünscht, kann die Anweisung WAIT Verwendung finden. Nach WAIT folgt der dezimale Wert einer Speicherstelle. Es sollte eine Speicherstelle sein, die nicht vom Programm anderweitig verwendet wird. Danach folgt hinter einem Komma ein Wert zwischen 0 und 255. Der Wert des Speicherplatzes wird mit dem folgenden Wert in der Anweisung verglichen. Ist die angegebene Speicherzelle nicht vorhanden oder ändert sich deren Wert nicht, begibt sich das Programm in eine Endlosschleife. Diese kann nur mit RESET verlassen werden. Dazu drücken Sie die CONTROL-Taste und anschließend den RESET-Knopf. Das Programm bleibt dabei erhalten.

## 4.33 Fehlerbehandlung

Der BASIC-Interpreter des Apple IIc ist mit einer Vielzahl von Fehlermeldungen ausgestattet. Sie erlauben es uns, jede Falscheingabe zu lokalisieren. Dabei handelt es sich meistens um syntaktische Fehler. Wird ein Wert zu hoch oder zu niedrig angegeben, erscheint eine Fehlermeldung. Bei der Ausführung des Programms wird die Schreibweise der Anweisungen und Funktionen überprüft. Ist diese falsch, meldet der Computer **SYNTAX ERROR.** Es können sich aber auch Fälle ergeben, die der Computer nicht interpretieren kann und darum falsch behandelt. Enthält z.B. eine Variable ein reserviertes Wort, dann wird die Anweisung falsch formuliert und kann in den wenigsten Fällen weiterbearbeitet werden.

Sie müssen als Programmierer bei der Vergabe von Variablennamen ganz genau aufpassen. Dialogprogramme sind schon bei der Programmierung so zu schützen, daß keine falsche Eingabe möglich ist. Bei INPUT-Anweisungen sollten Sie vorher definieren, wie eine Zahl oder ein Text auszusehen hat. Sie können das mit einer IF-THEN-Bedingung hervorragend realisieren. ON-GOSUB-Anweisungen sind hierbei auch dienlich. Sind diese Möglichkeiten jedoch nicht anwendbar, verwenden Sie einen neuen BASIC-Befehl.

**ONERR GOTO**

Mit dieser Anweisung ist es möglich, Fehler, die der BASIC-Interpreter erkannt hat, zu unterdrücken und in ein Unterprogramm zu verzweigen. Im Unterprogramm kann auf den Fehler hingewiesen werden. Mit der Anweisung

**RESUME**

wird zum Ausgangspunkt zurückgesprungen. RESUME hat dieselbe Wirkung wie RETURN bei einer GOTO-Anweisung. Soll das Programm an einer anderen Stelle fortfahren, so ist eine GOTO-Anweisung in das Unterprogramm einzufügen.

Beispiel:

```
10 ONERR GOTO 40
20 INPUT A
30 PRINT A
40 PRINT "Die Gleitkommazahl liegt au
ßerhalb des zulässigen Bereichs !"
50 RESUME
RUN

?12345678E99
Die Gleitkommazahl liegt außerhalb des
zulässigen Bereichs !
?
```

Mit

**PEEK (222)**

läßt sich der Errorcode ermitteln. So kann PEEK (222) in eine Variable gefaßt werden. Erscheint nun eine Fehlermeldung, dann kann ein Unterprogramm den Errorcode feststellen und eine individuelle Fehlermeldung ausgeben.

Beispiel:

```
10 ONERR GOTO 100
.
.
.
100 X = PEEK (222)
110 IF X = 16 THEN PRINT"Falscher Befehl !"
120 IF X = 69 THEN PRINT"Zahl zu groß !"
130 IF X = ......usw.
140 RESUME
```

Tabelle 4-1: Fehlercodes

| Error Code | Fehlermeldung des Applesoft-Interpreters |
|---|---|
| 0 | ?NEXT WITHOUT FOR ERROR |
| 16 | ?SYNTAX ERROR |
| 22 | ?RETURN WITHOUT GOSUB ERROR |
| 42 | ?OUT OF DATA ERROR |
| 53 | ?ILLEGAL QUANTITY ERROR |
| 69 | ?OVERFLOW ERROR |
| 77 | ?OUT OF MEMORY ERROR |
| 90 | ?UNDEFINED STATEMENT ERROR |
| 107 | ?BAD SUBSCRIPT ERROR |
| 120 | ?REDIM'D ARRAY ERROR |
| 133 | ?DIVISION BY ZERO ERROR |
| 163 | ?TYPE MISMATCH ERROR |
| 176 | ?STRING TOO LONG ERROR |
| 191 | ?FORMULA TOO COMPLEX-ERROR |
| 224 | ?UNDEFINED FUNCTION ERROR |
| 254 | (Fehlerhafte INPUT-Anweisung) |
| 255 | (Interrupt durch CTL-C) |

Im Anhang A sind die Fehlermeldungen des BASIC-Interpreters zusammengefaßt. Alle Fehlercodes sind mit PEEK (222) abrufbar und können in einer ONERR-Funktion Verwendung finden.

## 4.34 Joystick und Paddleansteuerung

Für alle, die noch nicht wissen, was ein Joystick oder Paddle ist, sei es hier nochmals erklärt. Als Joystick bezeichnet man einen Steuerhebel, der analoge Werte an den Computer übertragen kann. Ein Paddle oder Drehregler kann nur einen Wert übertragen, ein Joystick dagegen zwei. Besonders für die Steuerung von Spielen ist dies eine gute Anwendung. Der Joystick wird dazu verwendet, vertikale oder horizontale Steuerungen auf dem Bildschirm zu übernehmen. Um die Werte des Joysticks oder Paddles abfragen zu können, sind im Computer vier Speicherzellen reserviert, die immer die Werte bereithalten. Wir brauchen sie nur noch abzurufen, und schon können wir sie in unseren Programmen verwenden. Dazu gibt es den Befehl

**PDL**

Das ist die Abkürzung von Paddle. Denn jeder Speicherplatz kann nur einen Wert beinhalten. Es können somit vier Paddles oder zwei Joysticks angesprochen werden.

Beispiel:

```
5 HOME
10 HTAB 10: VTAB 10: PRINT PDL (0)
20 HTAB 10: PRINT PDL (1)
30 HTAB 10: PRINT PDL (2)
40 HTAB 10: PRINT PDL (3)
50 GOTO 10
```

Starten Sie dieses Programm, so erscheinen auf dem Bildschirm vier Zahlenwerte, die untereinander stehen. Diese Zahlen stellen die Positionswerte der Steuerhebel dar. Wird ein Joystick bewegt, dann ändern sich die Zahlenwerte auf dem Bildschirm. Ist kein Steuergerät angeschlossen, zeigen alle vier Zahlen den Wert 255. Im Normalfall liegt der Wert bei angeschlossenem Steuergerät zwischen 0 und 255.

## 4.35 Selektion von Peripheriegeräten

Haben wir z.B. einen Drucker an den Apple IIc angeschlossen, muß auch eine entsprechende Ansteuerung des Rechners erfolgen. Dazu gibt es zwei Befehle:

**IN#**

**PR#**

Mit IN# wird die Steckverbindung selektiert, deren Nummer der IN#-Anweisung folgt. Die Steckverbindungen sind von 0 bis 7 durchnumeriert. Wird eine Steckverbindung selektiert, so fließen alle Ausgabedaten zu dem Gerät, das an dieser Steckverbindung angeschlossen ist. Mit PR#1 wird zum Beispiel der Drucker ausgewählt.

Beispiele:

PR#3 selektiert die 80-Zeichen-Darstellung

PR#6 startet die Diskette von neuem

IN#6 startet ebenfalls die Diskette neu

Bei der Druckersteuerung im Kapitel 6 werden Sie noch mehr über die Anweisung PR# erfahren.

## 4.36 Funktionen

Etwas abweichend von Anweisungen sind Funktionen. Sie können Zahlenwerte, aber auch Werte, die in Variablen stehen, beeinflussen. Dabei gilt, daß Funktionen überall da eingesetzt werden können, wo Variable oder Konstante stehen. Ausgenommen links vom Gleichheitszeichen. Wir wollen im folgenden Abschnitt die Funktionen genauer betrachten. Sie sind eingeteilt in numerische Funktionen, Text-Funktionen und systembezogene Funktionen. Sie bestehen aus Funktionsbezeichnung und Argument. Das Argument kann aus einer Zahl oder mehreren Zahlen und Variablen bestehen.

### 4.36.1 Numerische Funktionen

Funktionen, die Zahlenwerte und numerische Variable beeinflussen, nennt man numerische Funktionen. Bei komplexen mathematischen Berechnungen sind sie unentbehrlich. Die wichtigsten Vertreter von mathematischen Funktionen sind die trigonometrischen, die Wurzelfunktion und die Exponente.

### 4.36.2 ABS

Diese Funktion ergibt den Betrag, der als Argument auf diese folgt. Dabei werden Argumente mit negativen Vorzeichen in positive Zahlen umgewandelt. ABS kann durch eine PRINT-Anweisung ausgegeben werden.

Beispiel:

```
10 PRINT ABS(12+12)
RUN
24


10 A=ABS (-12)
20 PRINT A
RUN
12
```

### 4.36.3 RND

Um Zufallszahlen erzeugen zu können, benötigen Sie die Funktion RND. In Integer-BASIC muß die Zahl, die in Klammern der Funktion folgt, zwischen -32767 und 32767 liegen. Bei Überschreitung dieser Werte erscheint die Fehlermeldung

***>32767

Die Zufallszahl liegt in Integer-BASIC immer zwischen 0 und der in Klammer angegebenen Zahl. Dabei wird die 0 mit angezeigt. In Applesoft-BASIC ist die Zufallszahl immer kleiner als 1, jedoch größer oder gleich 0. Wird die Zufallszahl erneut abgefragt, erscheint eine neue Zahl. Ist die Zahl negativ, wird immer dasselbe Ergebnis erscheinen. Bei negativen Zahlen ändert sich die Zufallszahl nur dann, wenn die Zahl in Klammern verändert wird.

**Beispiel in Applesoft:**

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 A = RND (12) | (z.B.) |
| 20 PRINT A | .289543246 |
| 30 GOTO 10 | .702887167 |
| RUN | .432895302 |
| | .302706239 |
| | usw. |

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 A = RND (-12) | 4.48235369E-08 |
| 20 PRINT A | 4.48235369E-08 |
| 30 GOTO 10 | 4.48235369E-08 |
| RUN | 4.48235369E-08 |
| | 4.48235369E-08 |
| | 4.48235369E-08 |
| | 4.48235369E-08 |
| | usw. |

Sie sehen, daß die Definition einer negativen Zahl in einer RND-Funktion immer den gleichen Zahlenwert ergibt.

**Beispiel in Integer-BASIC:**

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 A = RND (12) | 4 |
| 20 PRINT A | 4 |
| 30 GOTO 10 | 8 |
| RUN | 4 |
| | 7 |
| | 3 |
| | 6 |
| | 8 |
| | usw. |

| **Eingabe:** | **Ausgabe am Bildschirm:** |
|---|---|
| 10 A = RND (-12) | -5 |
| 20 PRINT A | -6 |
| 30 GOTO 10 | 0 |
| RUN | -1 |
| | -2 |
| | -8 |
| | -11 |
| | usw. |

In Integer-BASIC werden die Zufallszahlen immer in ganzen Zahlen ausgegeben. Dabei entscheidet die definierte Zahl, welches Vorzeichen die Zufallszahl hat. Ist die definierte Zahl positiv, so werden alle Zahlen positiv ausgegeben. Ist die definierte Zahl negativ, sind auch die Zufallszahlen negativ - außer 0.

### 4.36.4 SGN

Die Funktion dient zur Ermittlung des Vorzeichens einer angegebenen Zahl. Sie kann sowohl in Applesoft- als auch in Integer-BASIC eingesetzt werden. SGN muß einer Variablen zugeordnet sein. Ist der Wert, der in Klammern der Funktion SGN folgt, negativ, wird der Variablen der Wert -1 zugeordnet. Ist er positiv, so erhält die Variable den Wert 0.

**Beispiel in Applesoft:**

```
10 X = SGN (89.0234 - 102.89)
20 PRINT X
RUN

-1
```

**Beispiele in Integer-BASIC:**

```
10 A = SGN (-123)
20 PRINT A
30 END
RUN

-1
```

```
10 A = SGN (123)
20 PRINT A
30 END
RUN

1
```

```
10 A = SGN (10-10)
20 PRINT A
30 END
RUN

0
```

Hat die Berechnung in Klammern den Wert 0 oder ist der Wert 0, dann wird auch der Variablen von SGN der Wert 0 zugeordnet.

**ACHTUNG!** Die folgenden numerischen Funktionen können nur in Applesoft verwendet werden.

### 4.36.5 SIN

Die Funktion berechnet den Sinus der in Klammern angegebenen Zahl oder den Wert der in Klammern stehenden Berechnung. Das Argument muß im Radian angegeben werden. Besonders bei Kreisberechnungen wird diese Funktion eingesetzt.

Beispiel:

```
10 ? SIN (12.123)
RUN
-.428986606
```

Diese Funktion kann auch in der Betriebsart Einzelanweisung angewendet werden.

### 4.36.6 COS

Die Funktion berechnet den Cosinus der in Klammern angegebenen Zahl oder den Wert der in Klammern stehenden Berechnung. Das Argument muß in Radian angegeben werden.

Beispiel:

```
10 A=12.123
20 PRINT COS (A)
RUN

.903310849
```

Auch diese Funktion ist in der Betriebsart Einzelanweisung anwendbar.

### 4.36.7 TAN

Diese trigonometrische Funktion berechnet den Tangens des Arguments, das in Klammern hinter der Funktion steht. Das Argument muß in Radian angegeben werden.

Beispiel:

```
10 A = COS(12)
20 PRINT TAN(A)
RUN

1.12432048
```

### 4.36.8 ATN

Auch diese trigonometrische Funktion berechnet einen Radian-Wert der in Klammern stehenden Variablen, Berechnungen oder Zahlenwerte. Das Ergebnis ist der Arcustangens.

Beispiel:

```
10 PRINT ATN (123.34567)
20 PRINT ATN (12)
RUN

1.56268921
1.4876551
```

### 4.36.9 EXP

Die Funktion EXP hat folgende Syntax:

EXP (Argument)

EXP berechnet e hoch Argument. Dabei ist e eine numerische Konstante mit dem Wert e = 2.71828183.

Beispiel:

```
PRINT EXP(10)
22026.4658
```

Es entspricht der mathematischen Schreibweise $e^{10}$

### 4.36.10 INT

Diese Funktion darf nicht mit der Anweisung INT verwechselt werden. Richtig, beide werden gleich geschrieben, aber nur eine hat ein Argument in Klammern dahinter. Die INT-Anweisung, die auf Integer-BASIC umschaltet, kann nur in der Betriebsart Einzelan-

weisung benutzt werden. INT als Funktion ist meistens einer Variablen zugeordnet oder steht hinter einer PRINT-Anweisung. Die Funktion INT wandelt Zahlen mit Dezimalstellen hinter dem Komma in ganze Zahlen um.

Beispiel:

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| (Integer-Basic wurde geladen) | |
| INT | > |
| FP | Ü |
| ?INT(12.123) | 12 |
| (Integer-Basic ist nicht vorhanden) | |
| INT | ?SYNTAX ERROR |
| 10 A = 34.56784 | |
| 20 ? INT (A) | |
| RUN | 34 |

Probieren Sie es selbst am Computer einmal aus. Sie werden dieselben Ergebnisse erhalten.

### 4.36.11 LOG

Diese Funktion ermittelt den natürlichen Logarithmus des Arguments, das der Funktion in Klammern folgt. Ist das Argument gleich 0 oder eine negative Zahl, dann wird folgende Fehlermeldung ausgegeben:

**?ILLEGAL QUANTITY ERROR**

Beispiel:

```
PRINT LOG (1234)
7.11801621
```

**SQR**

Mit dieser Funktion können Sie die Quadratwurzel eines Arguments ermitteln. Diese muß, wie bei anderen Funktionen, in Klammern stehen.

Beispiel:

```
PRINT SQR(23)
4.79583153
```

**Text-Funktionen**

Text-Funktionen sind unentbehrlich bei der Programmierung von Dialogprogrammen, Bildschirmdarstellungen und Textverarbeitung. Wie schon bei den numerischen Funktionen, so sind auch hier die zu ermittelnden Werte in Klammern gesetzt. Die Text-Funktionen ASC und LEN können in Applesoft-BASIC und Integer-BASIC angewendet werden. Alle anderen Text-Funktionen bleiben jedoch nur in Applesoft-BASIC erhalten.

**4.36.12 ASC**

Die Text-Funktion ASC ermittelt den ASCII-Code des Zeichens, das in Klammern steht. Sind mehrere Zeichen in der Klammer, wird der Wert des ersten Zeichens ermittelt. Wird der ASCII-Code 0 ermittelt, erscheint die Fehlermeldung

**?ILLEGAL QUANTITY ERROR**

Beispiel:

```
10 A$ = "A"
20 PRINT ASC(A$)
RUN

65
```

### 4.36.13 LEN

Diese Text-Funktion errechnet die Länge eines Textes. Dabei muß der Text in eine Textvariable gefaßt sein. Leerzeichen und Steuerzeichen werden mitgezählt. Ist der Umfang des Textes länger als 255 Zeichen, erfolgt die Fehlermeldung

**?STRING TOO LONG ERROR**

Diese Möglichkeit tritt dann auf, wenn mehrere Textvariable zusammengefaßt wurden.

Beispiel:

```
10 A$ = "123456789012345678"
20 ? LEN (A$)
RUN

18


10 A$ = "12345678901234567890"
20 B$ = A$+A$+A$+A$
30 PRINT LEN (B$)
RUN

80
```

**ACHTUNG** ! Alle folgenden Text-Funktionen können nur in Applesoft angewendet werden.

### 4.36.14 CHR$

gibt das Zeichen aus, das dem angegebenen ASCII-Code entspricht. Dabei wird ein Wert zwischen 0 und 255 eingesetzt. Mit dieser Funktion ist es möglich, Steueraufgaben an die Peripheriegeräte zu übertragen. So kann die Funktion ein Diskettenlaufwerk, die Maus oder einen Drucker steuern. Alle Zeichen, die nicht über die Tastatur eingegeben werden, sind mit dieser Funktion aufrufbar.

Beispiel:

```
PRINT CHR$ (86)
V
```

Bei der Besprechung der Diskettenverwaltung wird diese Funktion praktisch eingesetzt.

### 4.36.15 LEFT$

Syntax von LEFT$:

```
LEFT$ (Textvariable$,Anzahl der Zeichen)
```

Die Text-Funktion zeigt so viele Zeichen von einer vordefinierten Textvariablen an, wie dem Wert in Klammern hinter dem Komma entsprechen. Dieser Wert muß zwischen 0 und 255 liegen. LEFT$ zählt dabei von links die Anzahl der Zeichen und gibt diese aus. Ist die Anzahl der Zeichen höher angelegt als definiert wurde, dann wird der ganze Text der Textvariablen ausgegeben. An mehreren Beispielen sollen diese Zusammenhänge klarwerden.

Beispiel:

```
10 A$ = "ABCDEFGHIJKLMNOP"
20 PRINT LEFT$ (A$,6)
RUN

ABCDEF
```

```
10 A$ = "ABCDEFGHIJ"
20 PRINT LEFT$ (A$,260)
RUN

ABCDEFGHIJ
```

### 4.36.16 RIGHT$

Wie schon bei LEFT$, wird auch hier ein Textteil ermittelt. Jetzt jedoch die letzten Zeichen. Dabei darf die Textvariable ebenfalls 255 Zeichen nicht überschreiten. Es werden Leerzeichen und Steuerzeichen gezählt und eventuell ausgegeben. An zwei Beispielen soll dies gezeigt werden.

Beispiel:

```
10 A$ = "A B C D(CTRL-G)"
20 PRINT RIGHT$(A$,4)
RUN

C D ("BEEP" (Das Klingelzeichen
      wurde mit eingefügt))
```

```
10 A$ = ""
20 PRINT RIGHT$(A$,3)
RUN

("BEEP" "BEEP" "BEEP")  A$ = 3x CTRL-G
```

### 4.36.17 MID$

Um einen Textteil aus einer Textvariablen herauszulösen, hatten wir bereits die Funktionen LEFT$ und RIGHT$ betrachtet. Es ist aber auch möglich, Textsegmente auszulagern und separat auszugeben. Die Funktion MID$ stellt uns diese Möglichkeit zur Verfügung. MID$ verlangt als Angabe den Platz, ab welchem Zeichen wie

viele Zeichen ausgegeben werden. Außerdem muß die Definition der Textvariablen angegeben sein.

Syntax von MID$:

MID$ (Textvariable$, welche Stelle, wieviel Zeichen)

Die Textvariable sollte zwischen 1 und 255 Zeichen beinhalten. Wird diese Grenze überschritten, erfolgt eine Fehlermeldung. Auch die Angaben in Klammern dürfen den Wert 255 nicht überschreiten. Wird die Anzahl der auszugebenden Zeichen größer definiert als Zeichen in der Textvariablen vorhanden sind, wird kein Text ausgegeben. Genauso verhält es sich, wenn die Stelle, ab der ausgegeben wird, den Umfang der Textvariablen überschreitet.

Beispiel:

```
10 A$ = "ABCDEFGHIJKLMNOPQRSTU"
20 PRINT MID$ (A$,6,8)
RUN

FGHIJKLM
```

### 4.36.18 VAL

Mit dieser Funktion können Textvariable in einen Zahlenwert umgewandelt werden. Alle Zeichen, die in der Textvariablen angegeben sind, werden auf ihre Art überprüft. Dabei wird allen Buchstaben, außer dem Großbuchstaben E, der Wert 0 zugeordnet. Die Ziffern 0 bis 9 sowie die Sonderzeichen Punkt, Plus- und Minuszeichen und das Leerzeichen sind als numerische Werte definiert. Alle anderen Zeichen wird ebenfalls der Wert 0 zugeordnet. Ist die Textvariable länger als 255 Zeichen, erfolgt die Fehlermeldung **?STRING TOO LONG ERROR.**

Beispiel:

```
10 A$ = "  456 +-.E"
20 PRINT VAL (A$)
RUN
456
```

```
10 A$ = "A456ABCE"
20 PRINT VAL A$
RUN
0
```

Wie Sie an den Beispielen sehen können, sind alle Buchstaben und Sonderzeichen ignoriert worden. Wird am Anfang der Textvariablen ein Buchstabe angetroffen, erhalten wir das Ergebnis 0.

### 4.36.19 STR$

Mit dieser Funktion wird eine Berechnung in einen Text umgeformt. Es ist derselbe Vorgang, als ob eine Berechnung mit PRINT ausgegeben wird. Auch diese Textfunktion ist nur in Applesoft-BASIC anwendbar. Wird der festgesetzte Wert einer Gleitkommazahl überschritten, erscheint die Fehlermeldung

**?OVERFLOW ERROR**

Beispiel:

```
PRINT STR$(12/23)
.52173913

? STR$ (1234567890)
1.23456789E+09
```

### 4.36.20 Definition von numerischen Funktionen

In Applesoft-BASIC können zu den vorhandenen numerischen Funktionen weitere Funktionen definiert werden. Dazu verwenden wir die Befehlsfolge

**DEF FN**

Diese Möglichkeit sollte jedoch nicht überbewertet werden. Sie können damit keine komplizierten Funktionen definieren. Der Buchstabe oder das Zeichen, das nach FN folgt, wird dann als Funktionsname eingesetzt. Steht hinter FN AA, dann heißt die neue Funktion FN AA (X). X wird als Platzhalter eingesetzt.

Beispiel:

```
10 DEF FN X(Y) = 20 * Y
20 Y = 20
30 PRINT FN X(Y)
```

Die Variable Y, die in Klammern steht, wird als Maskenvariable eingerichtet. Kommt die Funktion zur Ausführung, dann sind die Werte, die bei der Definition angegeben wurden, maßgebend.

### 4.36.21 Systembezogene Funktionen

Im folgenden Abschnitt wollen wir uns mit Funktionen oder Anweisungen beschäftigen, die ein Bindeglied zwischen BASIC und Maschinensprache herstellen. Die Befehle PEEK, POKE und CALL lassen einen Blick in die Speicherstellen des Computers zu. Die Anweisungen erlauben es, den Inhalt einer Speicherstelle zu ermitteln, in eine Speicherstelle zu schreiben und Maschinenunterprogramme aufzurufen.

### 4.36.22 PEEK

PEEK läßt einen Blick in die Speicherzellen des Computers zu. Die Funktion ist als Einzelanweisung oder in einem Programm einsetzbar. Wie schon bei anderen Funktionen, so wird auch hier die abzufragende Speicherstelle dezimal, in Klammern, eingegeben. Der Wert liegt zwischen 0 und 65535 oder -1 und -65535 in Applesoft-BASIC. In Integer-BASIC sind Werte zwischen -32767 und 32767 einzugeben. Da Applesoft-BASIC den kompletten Speicherbereich mit den Zahlen von 0 bis 65535 abdeckt, sind die negativen Zahlen die Werte, die nach Abzug von 65536 vom positiven Wert der Speicherzelle überbleiben. In Integer-BASIC kann mit -32767 bis 32767 ebenfalls der ganze Speicherbereich abgefragt werden.

**Beispiel in Applesoft:**

```
? PEEK (-65535) ergibt denselben Wert wie
? PEEK (1)
```

**Beispiel in Integer-BASIC:**

```
>PRINT PEEK (-300)
12
>PRINT PEEK (300)
0
```

Setzen Sie hinter PEEK eine Variable, dann ist es möglich, den Wert einer bestimmten Speicherstelle immer wieder abzufragen. Speicherstellen des Festwertspeichers sind damit offen und lesbar.

### 4.36.23 POKE

Wird in einem BASIC-Programm gewünscht, daß sich Speicherzellen im Computer verändern sollen, ist die Funktion POKE anzuwenden. Damit können Sie Maschinenprogramme, die als DATA-Anweisungen im BASIC-Programm abgelegt sind, in einen bestimmten Speicherbereich legen. Unterprogramme in Maschinensprache sind damit leicht einzulesen und anschließend im BASIC-Programm immer verfügbar.

Beispiel:

```
POKE 300,34
PRINT PEEK (300)
34
```

In Speicherstelle dezimal 300 wurde der dezimale Wert 34 geschrieben. Gelesen wird diese Speicherstelle mit PEEK. Dabei ist zu beachten, daß Speicherzellen, die im Bereich des Festwertspeichers liegen, nicht verändert werden. Der eingegebene Wert muß zwischen 0 und 255 liegen.

### 4.36.24 HIMEM, LOMEM

Diese Anweisungen teilen dem BASIC-Speicherbereich neue Grenzen zu. Das kann bei großen Maschinenunterprogrammen und grafischen Darstellungen von Nutzen sein. HIMEM gibt den oberen und LOMEM den unteren Bereich an. Um den verfügbaren Speicherplatz zu ermitteln, benutzen wir die Funktion

**FRE**

Sie errechnet den freien Speicherplatz für BASIC-Programme und Daten. Außerdem löscht FRE alte Variableninhalte, deren Variablen ein neuer Inhalt zugeordnet wurde. Besonders bei Behandlung von vielen Textvariablen kommt uns diese Funktion zugute. Wird FRE ein negativer Wert zugeordnet, dann muß 65536 hinzuaddiert werden.

Beispiel:

```
PRINT FRE (0)
- 18435

PRINT -18435 + 65536
47101
```

Bei diesem Beispiel sind fast 46 KByte frei. Denn 47101:1024 = 45,9970703. (1024 Byte entsprechen 1KByte.) Jetzt wollen wir den Speicherbereich für BASIC-Programme einschränken.

- 18435 ist der verfügbare Speicherbereich

```
Eingabe:    HIMEM: 30000
            LOMEM: 15000

            PRINT FRE(0)
            15000
```

Wir haben noch 15000 Speicherplätze frei verfügbar für BASIC. In Integer-BASIC sind die Anweisungen HIMEM und LOMEM nur in der Betriebsart Einzelanweisung anwendbar. Die Funktion FRE steht uns nur in Applesoft-BASIC zur Verfügung.

### 4.36.25 CALL

Haben wir Daten, die Maschinenunterprogramme darstellen, aus dem BASIC-Programm in den internen Arbeitsspeicher geladen, ist auch das Starten in BASIC notwendig. CALL läßt ein Maschinenprogramm ab der Stelle ablaufen, die dezimal hinter CALL angegeben wird. Dieser Wert darf nur zwischen 0 und 65535 oder -1 und -65535 liegen. Um Programme, die bereits im Festwertspeicher abgelegt sind, starten zu können, ist ebenfalls die Anweisung CALL nötig.

Beispiel:

CALL -198 gibt einen TON aus

CALL -936 löscht den Bildschirm wie die Anweisung HOME

Die Zahl, die hinter CALL angegeben wird, ist der dezimale Wert der Startadresse im internen Speicher. Im Anhang D sind Unterprogramme angegeben, die sich mit CALL starten lassen.

### 4.36.26 USR

Genau wie CALL leitet auch USR einen Sprung zu einem Unterprogramm ein. Dabei müssen aber in den Speicherstellen dezimal 10, 11 und 12 die Sprungbefehle abgespeichert sein, die zum Unterprogramm zeigen. Das geschieht mit

```
POKE 10,..
POKE 11,..
POKE 12,..
```

In Klammern steht hinter der USR-Funktion die Variable, die USR zugeordnet wird. USR fragt den Wert im Akkumulator des internen Speichers ab.

Beispiel:

```
10 POKE 10,76:POKE 11,0:POKE 12,32
20 FOR A=8192 TO 8198
30 READ B
40 POKE A,B:NEXT A
50 FOR A = 1 TO 10
60 ? A,USR(A)
70 NEXT A
80 DATA 165,157,105,10,133,157,96
```

Bei diesem Beispiel wird im Unterprogramm, das bei Speicherstelle $2000 beginnt, ein Betrag zum Akku addiert. Nach dem Verlassen des Unterprogramms wird das Ergebnis ausgegeben.

## 4.37 Der Programmtest

Besonders bei sehr langen Programmen können sich Fehler eingeschlichen haben. Im folgenden Abschnitt wollen wir Möglichkeiten erarbeiten, mit denen Sie Ihre Programme auf Herz und Nieren testen können.

### 4.37.1 Die modulare Programmierung

BASIC-Programme, die mehrere hundert Programmzeilen aufweisen, sind meist unübersichtlich und schwer in ihrer Gesamtheit zu überprüfen. Zerlegen Sie das Programm in kleine Teile, und testen Sie jedes Teil für sich. Danach kann das Programm zu einer Einheit zusammengefaßt werden. Dazu dienen Ihner die Anweisungen END oder STOP. Fügen Sie diese Befehle am Ende Ihrer einzelnen Programmbereiche ein, dann können Sie das Programm teilen, ohne wirklich Programmzeilen zu löschen.

GOTO-Anweisungen eignen sich hervorragend, um das Programm in einen anderen Programmteil zu lenken. Die eingefügten Anweisungen sind nach der Testphase wieder herauszunehmen. Darum sollte jede eingefügte Anweisung peinlichst genau aufgeschrieben werden. Eine zurückgelassene Testanweisung kann das Programm stören.

### 4.37.2 Einfügen von PRINT-Anweisungen

Um die Werte von Variablen zu testen, können diese durch Einfügen einer PRINT-Anweisung abgefragt werden. Hat eine bestimmte Variable eine besondere Bedeutung, ist nach der Abarbeitung der Definition eine PRINT-Anweisung zu setzen, die die Variable beinhaltet. Wird das Programm gestartet, zeigt PRINT den Wert der Variablen am Bildschirm an. Das Ergebnis von Berechnungen, das einer Variablen zugeordnet ist, wird auf diese Weise abrufbar, ohne den Programmablauf zu behindern.

In Integer-BASIC kann die Anweisung **DSP** eingefügt werden. Sie ist nur für numerische Variable zu verwenden. Nach der Anweisung DSP wird der Variablenname angegeben, der aufgezeigt werden soll. Es ist anzuraten, daß diese Anweisung vor der Definition der Variablen gesetzt wird. Ändert sich der Wert der Variablen, wird der numerische Wert mit der Programmzeilennummer, in der die Änderung aufgetreten ist, am Bildschirm angezeigt. Der Befehl DSP wir durch die Anweisung **NODSP** annulliert. In der Betriebsart Einzelanweisung muß das Programm mit GOTO gestartet werden.

**Beispiel für die Einzelanweisung:**

```
DSP X GOTO 10
```

**Beispiel im Programm:**

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| >DSP A | #10 A=1 |
| >10 A = A + 1 | 1 |
| >20 PRINT A | #10 A=2 |
| >30 GOTO 10 | 2 |
| RUN | #10 A=3 .....usw. |

Wird das Programm mit RUN gestartet, löscht RUN die DSP-Anweisung und startet das Programm normal. Fügen Sie deshalb DSP mit einer eigenen Programmzeile in das Programm ein. Nun kann das Programm auch mit RUN gestartet werden.

### 4.37.3 TRACE, NOTRACE

Für die Überprüfung und das Austesten eines Programmablaufs dient die Anweisung **TRACE**. Geben Sie TRACE als Einzelanweisung ein, dann wird der Programmablauf am Bildschirm in Form von Programmzeilennummern angezeigt. Der Computer gibt die Zeilennummer an, die er Schritt für Schritt abarbeitet. Auch Sprunganweisungen werden somit sichtbar. Soll diese Funktion wieder ausgeschaltet werden, gibt man den Befehl **NOTRACE** ein. Mit der Anweisung SPEED kann die Ablaufgeschwindigkeit geregelt werden. So erleichtern Sie sich das Ablesen der Zeilennummer.

Beispiel:

| Eingabe: | Ausgabe am Bildschirm: |
|---|---|
| 10 A = A + 1 | #10 #20 1 |
| 20 PRINT A | #30 #10 #20 2 |
| 30 GOTO 10 | #30 #10 #20 3 |
| RUN | #30 #10 #20 4 .....usw. |

Damit wären wir am Ende der BASIC-Einführung angelangt. Sie sollten nun die Beispiele ausprobieren und selber experimentieren.

# 5

# Die Programmierung und Anwendung der Maus

## 5 Die Programmierung und Anwendung der Maus

Das folgende Kapitel soll Ihnen eine Einführung in das Programmieren der Maus in BASIC und die Anwendung mit Mouse-Paint zeigen. Die Maus haben wir schon bei der Besprechung der Pheripheriegeräte kennengelernt. Sie ist seit der Vorstellung der Apple LISA ein beliebtes Eingabemittel. Mit ihr ist es möglich, am Bildschirm schnell zu reagieren. Durch einfaches Hin- und Herbewegen und "Klicken" mit dem Taster kann eine interaktive Ein- und Ausgabe erfolgen.

### 5.1 Die Apple-Maus

Die Apple-Maus ist ein mechanisches, optisches und elektronisches Eingabeinstrument. Mit einer gummibeschichteten Kugel wird über eine Andruckrolle und zwei Gleitrollen die Bewegung übertragen. An den zwei Gleitrollen sind axial perforierte Kunststoffscheiben angebracht. Diese drehen sich in einer winzigen Lichtschranke. Bei einer Bewegung entstehen so fortlaufende Unterbrechungen, die als elektronische Impulse in die Regelelektronik des Computers geleitet werden. Dabei arbeiten zwei Lichtschranken für die X- und Y-Richtung.

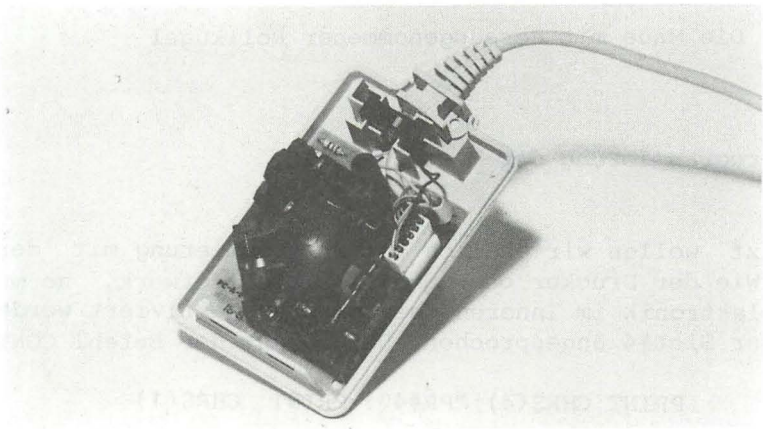


Abb. 5-1: Im Innern der Apple-Maus

Die Maus sollte immer auf einer glatten und sauberen Unterlage oder einem Schreibtisch bewegt werden. Trotzdem kann sich Staub und Schmutz im Inneren der Maus ablagern. Auch daran haben die Apple-Leute gedacht. Die Kugel kann nach dem Entfernen (drehen) der schwarzen Haltescheibe aus der Maus entnommen werden. So können Sie die Kugel und die Gleitrollen reinigen.

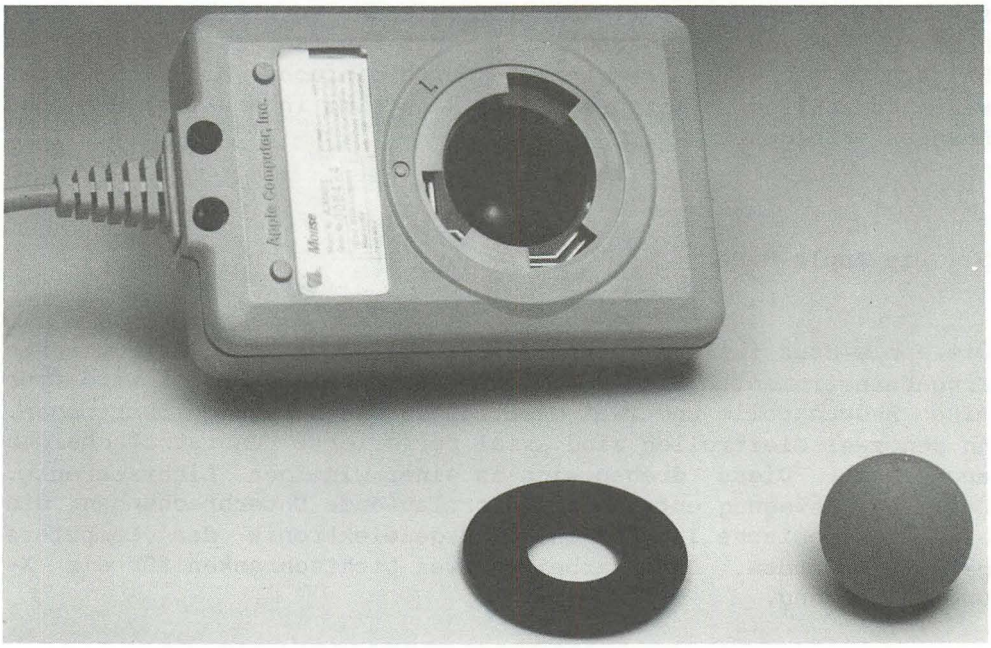



Abb. 5-2: Die Maus mit herausgenommener Rollkugel

## 5.2 Die Programmierung der Maus in BASIC

Aber jetzt wollen wir endlich zur Programmierung mit der Maus kommen. Wie der Drucker oder das Diskettenlaufwerk, so muß auch die Mauselektronik im Inneren des Computers aktiviert werden. Dazu wird der Slot#4 angesprochen. Dazu dient der Befehl CONTROL-D.

```
PRINT CHR$(4);"PR#4": PRINT  CHR$(1)
```

Damit wird der Steckplatz mit der Nummer 4 aktiviert, die Mauselektronik spricht an, und die Mausposition wird auf Null gesetzt. Jetzt müssen wir der Mauselektronik mitteilen, daß eine Ausgabe auf dem Bildschirm gewünscht wird.

```
PRINT CHR$(4);"PR#0"
```

PR#0 ist die SLOT-Nummer der Tastatur und wird jetzt auf den Bildschirm gelenkt. CHR$(4) steht für CONTROL-D. Jetzt werden drei Variable der Maus zugeordnet, an die sie ihre Position und ihren Schaltzustand meldet.

```
PRINT CHR$(4);"IN#4":INPUT " ";X,Y,S
```

Die Variablennamen könnten auch anders benannt werden. In unserem Beispiel wird der Variablen X der horizontale Wert, der Variablen Y wird der vertikale Wert zugeordnet. Die Variable S zeigt den Zustand des Schalters an. X und Y können einen Wert zwischen 0 und 1023 anzeigen. Dieser Wert wird immer positiv sein. Der Schalter mit der Variablen S hat einen Wert zwischen 1 und 4 und kann sowohl positiv als auch negativ sein. Im Abfragezustand ist er immer zwischen 1 und 4. Wird eine Taste auf der Tastatur betätigt, geht er in einen negativen Zustand.

Die Zahlen von 1 bis 4 haben folgende Bedeutung:

| | Tastenzustand zur Zeit | letzte Abfrage |
|---|---|---|
| 1 = | gedrückt | gedrückt |
| 2 = | gedrückt | frei |
| 3 = | frei | gedrückt |
| 4 = | frei | frei |

Wenn Sie das Beispiel aus dem Bedienerhandbuch eintippen, können Sie den Vorgang gut beobachten. Geben Sie vorher die Anweisung

**SPEED = 1**

ein. Jetzt läuft die Abfrage wesentlich langsamer. Sie können die Tastenwerte jetzt gut ablesen. Hier nochmals das Beispiel aus dem Apple-Mouse-IIc-User's-Manual.

```
10 HOME
20 PRINT "Das ist eine Maus-Demo"
30 PRINT CHR$(4); "PR#4": PRINT CHR$(1)
40 PRINT CHR$(4); "PR#0"
50 PRINT CHR$(4); "IN#4"
60 INPUT ""; X,Y,S
70 VTAB 10:PRINT X;"    ",Y"    ",S"  "
80 IF S > 0 THEN 60
90 PRINT CHR$(4); "IN#0"
100 PRINT CHR$(4); "PR#4": PRINT CHR$(0)
110 PRINT CHR$(4); "PR#0"
120 POKE -16368,0: REM LÖschen des Tastatursignals
130 END
```

Wie Sie ab Zeile 90 sehen können, wird die Maus wieder in den inaktiven Zustand zurückgesetzt. Die Ausgabe wird auf die Tastatur geleitet. Durch das Drücken einer beliebigen Taste auf der Tastatur geht die Variable S auf -4. Die Bedingung in Zeile 80 ist nicht erfüllt. Dadurch wird Zeile 90 eingeleitet. Der Cursor erscheint wieder, und wir befinden uns wieder im BASIC-Modus.

## 5.3 MousePaint

Beim Kauf einer Apple-Maus erhalten Sie eine Einführungs- und Anwendungsdiskette mit der Bezeichnung "MousePaint". In diesem Abschnitt wollen wir uns die Möglichkeiten der Maus am Beispiel dieser Software anschauen. Nach dem Starten der MousePaint-Diskette erscheint folgendes Bild auf dem Bildschirm:

**MousePaint**

**Press RETURN to learn how to use**
**the mouse, or**
**Press ESC to quit, or**

**Click the mouse button to go**
**directly to MousePaint**

**Copyright 1983 Apple Computer, Inc.**

Wenn Sie jetzt die RETURN-Taste drücken, können Sie einen kleinen Lehrgang mit der Maus absolvieren. Sie erfahren dort, was Symbole und Auswahlmenüs sind. Sie lernen den Umgang mit der Maus kennen. Diese Beispiele sind sehr hilfreich für das Erlernen der Mausbedienung. Wir wollen uns aber mit den Möglichkeiten von MousePaint beschäftigen. Dazu drücken wir den Taster an unserer Maus. Wir nennen diese Tätigkeit auch "klicken". Für die Leser, die noch keine Maus besitzen, sind die nächsten Seiten ein kleiner Überblick der Möglichkeiten, die die Maus bietet.

Wenn das MousePaint-Programm geladen ist, erscheint eine Tafel mit vielen Werkzeugen und Mustern. Das Arbeitsfeld ist noch leer, und der Mauszeiger steht in der oberen linken Bildschirmecke.

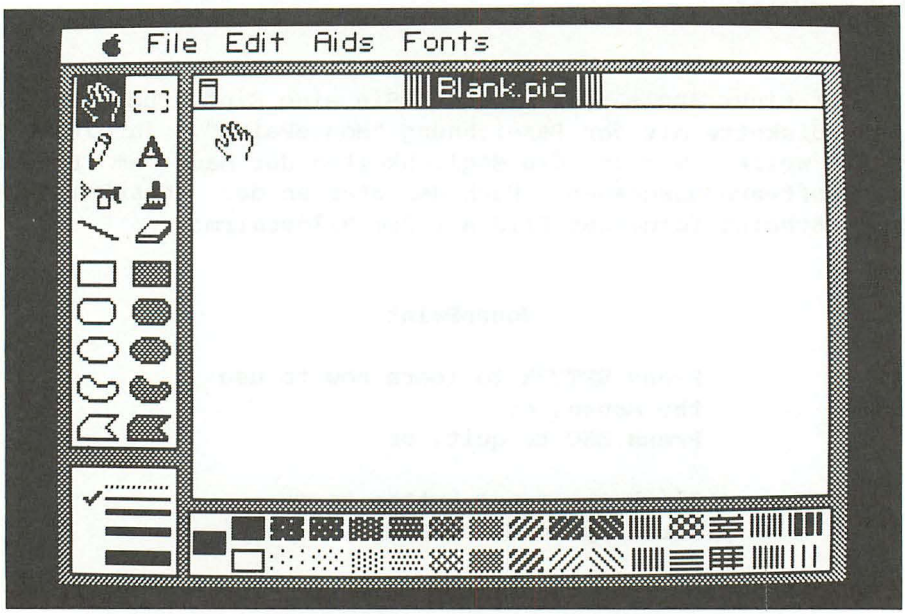


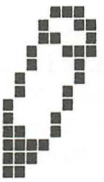

Abb. 5-3: Die MousePaint-Menüfelder

Um mit MousePaint arbeiten zu können, müssen wir ein neues "Zeichenblatt" hervorholen. Wir klicken auf das Wort FILE, und es erscheinen einige Menüpunkte. Sie müssen die Taste solange gedrückt lassen, bis Sie den richtigen Menüpunkt gefunden haben. Wenn Sie mit dem Pfeil nach unten wandern, färben sich die Felder, über denen der Zeiger steht, schwarz. Wir wählen das Feld "Blank HiRes Screen" aus. Lassen Sie jetzt den Taster los. Es erscheint ein weißes Arbeitsblatt mit der Überschrift "Blank.pic". Wird das Bild später abgespeichert, erhält es einen Namen. Das Bild hat nun beim Einladen den neuen Namen als Bildüberschrift. Wir wollen uns einige Werkzeuge, die in der linken Bildhälfte stehen, einmal genauer anschauen.

## 5.4 Werkzeuge in Mouse-Paint

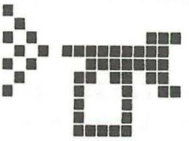
Die Hand hat die Aufgabe, das Bild zu verschieben. Was Sie auf dem Bildschirm sehen, ist nur ein Teil des ganzen Arbeitsblatts. Wenn Sie etwas zeichnen, dann können Sie mit Hilfe der Hand das Bild so verschieben, daß mehr Platz zur Verfügung steht.

Das gestrichelte Kästchen hat eine wichtige Bedeutung. Es kann Teile des Bildes aktivieren. Mit diesem Werkzeug können Duplikate von gezeichneten Bildteilen gemacht werden. Und es dient außerdem zum Löschen von Bildteilen. Wir werden noch auf dieses Instrument zurückkommen.

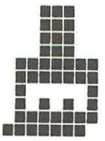


Der Stift kann zum Freihandzeichnen verwendet werden. Mit ihm können Sie wie mit einem Bleistift zeichnen. Er hat stets die gleiche Strichstärke. Sie kann nicht verändert werden, und die Farbe bleibt immer Schwarz.

Der Buchstabe A symbolisiert die Möglichkeit, in der Grafik mit der Tastatur zu schreiben. Dabei kann unter 5 Schriftarten gewählt werden. Im Menüpunkt FONTS sind diese Schriftarten aufgeführt. Es können alle Schriftarten auf einer Bildseite verwendet werden.

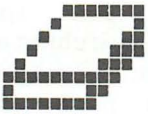


Sie können die Sprühdose wie in Wirklichkeit handhaben und damit Raster und Schatten sprühen. Die Farbe ist abhängig von dem gewählten Muster am unteren Bildschirmrand. Dabei kann ein vorhandenes Muster nicht neu übersprüht werden. Die Muster fließen ineinander, das Bild wird schwarz.



Der Pinsel ist ein weiteres Malwerkzeug. Seine Form kann im Menü Aids unter Menüpunkt "Set Brush" ausgewählt werden. Es stehen 16 verschiedene Pinselformen zur Verfügung. Der Pinsel zeichnet wie der Stift nur mit der Farbe Schwarz.

Der schräge Strich dient als Hilfsmittel, um gerade Striche zu erzeugen. Der Strich wird von dem Punkt aus gezogen, an dem zum ersten Mal geklickt wurde. Danach kann der Strich in alle Richtungen auf jede Länge gezogen werden: dabei muß die Taste gedrückt sein. Wird die Taste losgelassen, bleibt der Strich auf dem Arbeitsblatt in der Position zurück, in der er sich zuletzt befunden hat.



Das Symbol, das ausieht wie ein flacher Schuhkarton, ist ein elektronischer Radiergummi. Mit diesem Symbol können Sie Bildteile auslöschen. Sie brauchen dazu nur das Symbol auszuwählen. Wird die Taste gedrückt, ist der Radiergummi aktiviert und löscht die Bildteile, die er berührt.

In diesem Feld befinden sich noch einige weitere Grafiksymbole. Werden sie ausgewählt, kann z.B. ein Rechteck oder ein Kreis gestaltet werden. Ist das Symbol aktiviert, erscheint der Mauszeiger als Kreuz. Das Symbol kann geöffnet und vergrößert werden. Solange die Taste gedrückt bleibt, kann die Form in jede Richtung verändert werden. Lassen Sie die Taste los, bleibt die zuletzt gestaltete Form stehen. Die Symbole, die ausgefüllt sind (Abb.5-3), haben die gleiche Funktion, sind aber nach dem Loslassen der Maustaste mit dem Muster gefüllt, das ausgewählt wurde.

Wie Sie in Abbildung 5-3 sehen können, sind in der unteren linken Bildschirmecke verschiedene Strichstärken angegeben. Die jeweils vorher aktive Strichstärke ist mit einem kleinen Häkchen versehen. Das Strichsymbol und die anderen grafischen Symbole werden immer mit der aktivierten Strichstärke ausgeführt.

Die Muster in der unteren Bildschirmhälfte können ebenfalls mit dem Mauszeiger ausgewählt werden. Dabei zeigt das ganz links stehende Feld das aktivierte Muster an. Mit diesen Mustern werden die Grafiksymbole ausgefüllt, die gewählt werden.

## 5.5 Architekturplanung auf dem Apple IIc

An einem anschaulichen Beispiel sollen die Möglichkeiten von MousePaint gezeigt werden. So kann auch auf dem Apple IIc das erreicht werden, was auf dem Macintosh möglich ist. Um das folgende Beispiel nachvollziehen zu können, müssen Sie das Programm Mouse Paint laden.

Nachdem wir unser Arbeitsblatt geöffnet haben, wird jetzt das rechteckige Grafiksymbol ausgewählt. Mit diesem Symbol wollen wir einen Grundriß eines Zimmers zeichnen. Dazu klicken wir auf das Symbol und führen den Zeiger zur gewünschten Stelle. Das Symbol wird geöffnet und auf die richtige Größe gezogen. Damit erstelle ich jetzt auch die Fenster. Außerdem wird mit dem Strichsymbol eine offene Türe gezeichnet.

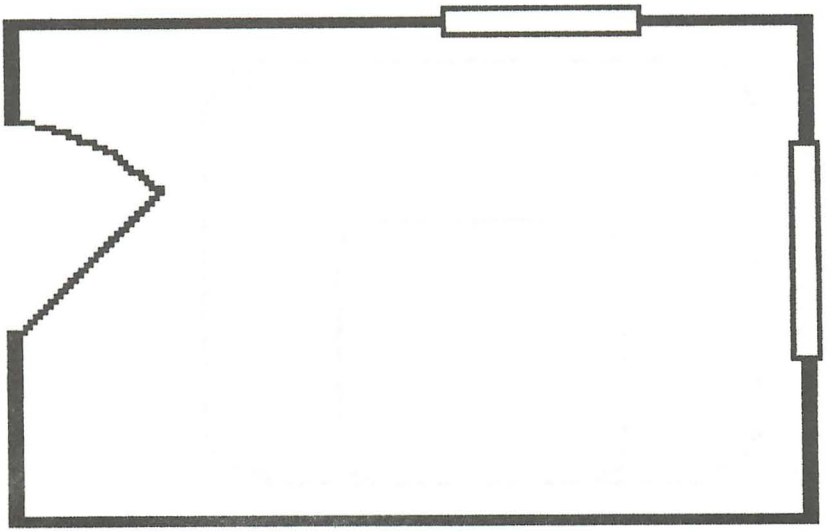


Abb. 5-4: Grundriß

Um zu sehen, wie die Raumeinteilung aussehen wird, müssen Möbel und Einrichtungsgegenstände eingezeichnet werden. Dazu entwerfen wir einige Möbelgrundrisse außerhalb des gezeichneten Zimmers. Wir wollen einen Polstersessel zeichnen. Dazu klicken wir das Reckteck mit den abgerundeten Ecken an. Das Symbol wird geöffnet und in die entsprechende Form gebracht. Ist das entworfene Symbol fertig, können wir es kopieren, denn es sollen zwei Sessel in das Zimmer gestellt werden. Klicken Sie das gestrichelte Recktecksymbol an, und aktivieren Sie den Sessel. Legen Sie dabei die gestrichelte Linie genau auf die Umrisse des Sessels. Das Koordinatenkreuz, in das sich der Zeiger verwandelt hat, ist dabei eine große Hilfe. Wird der Taster der Maus losgelassen, ist das Sesselsymbol aktiviert. Es kann mit dem Zeiger verschoben werden. Um eine Kopie anzufertigen, ist der Menüpunkt "Cut" aus dem Menü "Edit" auszuwählen. Das Sesselsymbol verschwindet und wird im internen Speicher abgelegt. Mit dem Menüpunkt "Paste" ausgewählt erscheint eine neue Hand, die mit dem Zeigefinger nach rechts zeigt. Klicken Sie einmal, erscheint unser Sessel wieder. Solange die Taste gedrückt ist, kann das Symbol verschoben werden.

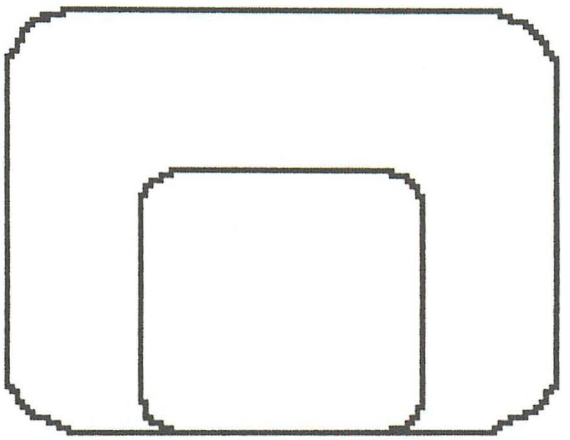


Abb. 5-5: Das neue Symbol

Mit dem Menüpunkt "Undo" wird das zuletzt eingesetzte Symbol wieder gelöscht. Auch mit "Copy" kann das Symbol in den Speicher gelegt werden. Wird "Paste" gewählt, erscheint das ausgeschnittene Symbol wieder. So kann man sich eine Menge Zeit sparen und muß nicht die gleichen Segmente öfters zeichnen. Im Menü "File" gibt es noch viele Punkte, die das Abspeichern und Aufrufen von Bildern beinhalten. Mit dem Menüpunkt "Print Picture", läßt sich das gesamte Arbeitsblatt auf dem Drucker anschließend ausdrucken.

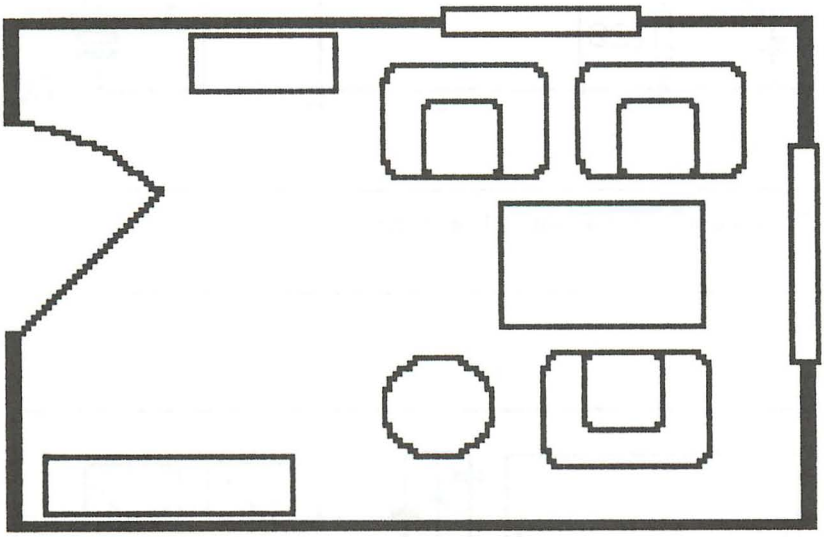


Abb. 5-6: Der Zeichenblattausdruck

Damit sind wir am Ende der Übungslektion von MousePaint angelangt. Ich habe versucht, Ihnen einen kleinen Schritt weiterzuhelfen. Sie können mit MousePaint experimentieren und kreativ tätig sein. Auf den nächsten Seiten finden Sie noch weitere Anwendungsmöglichkeiten, die mit MousePaint erstellt wurden.

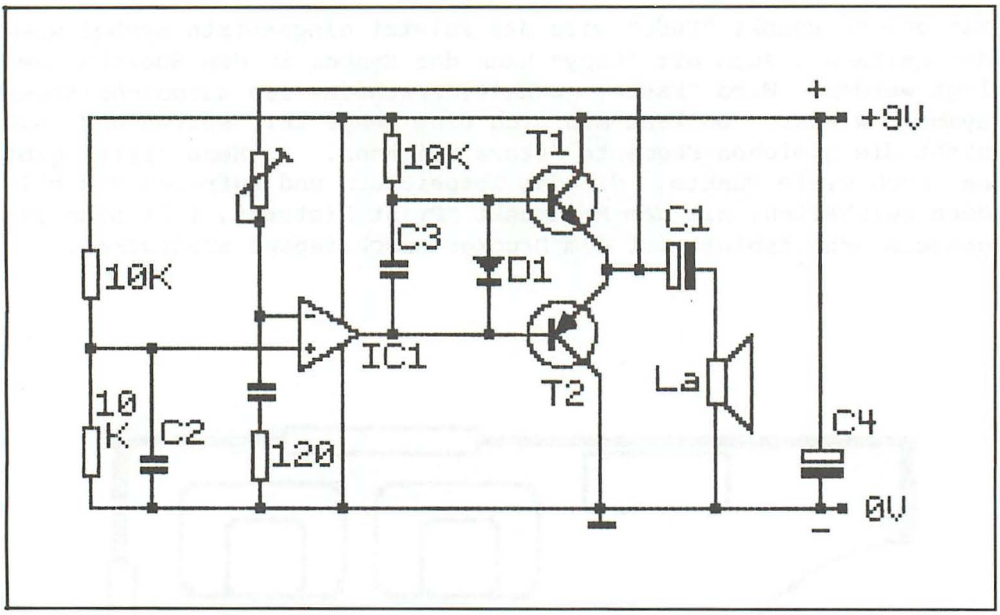



Abb. 5-7: Anwendung in der Elektronik

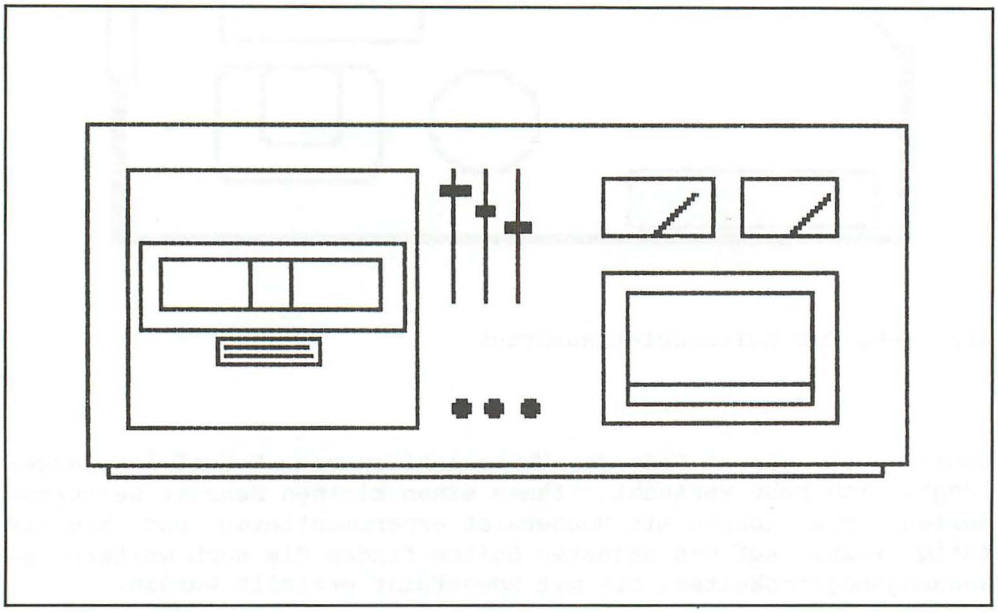


Abb. 5-8: Anwendung im Designentwurf

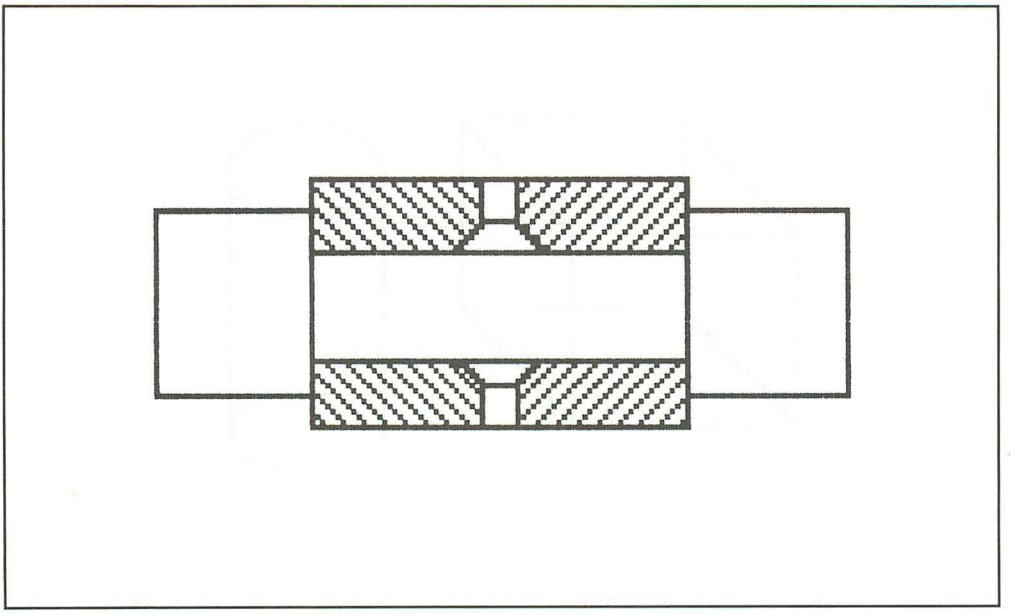


Abb. 5-9: Anwendung im Maschinenbau

Abb. 5-10: Anwendung in der Werbung

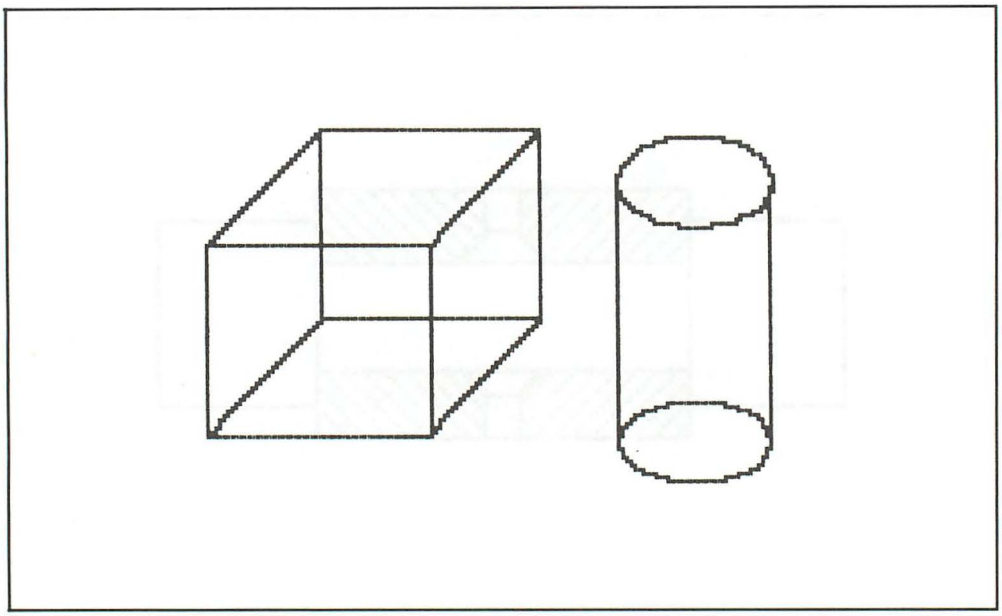


Abb. 5-11: Anwendung im Schulbereich

Abb. 5-12: Einladung mit MousePaint

# 6

# Der Druckerbetrieb unter BASIC

## 6 Der Druckerbetrieb unter BASIC

Am Beispiel des Druckers "Imagewriter" wollen wir die Druckersteuerung in der Betriebsart Direktanweisung und in Programmen näher betrachten.

Ein Drucker kann in vielerlei Hinsicht ein hilfreiches Mittel sein. Bei langen Programmen können die Listings über den Drucker ausgegeben werden, wodurch eine leichte Überprüfung möglich ist. Korrekturen kann man auf dem Druckerpapier protokollieren und später im Computer abändern. Werden Daten mit dem Computer erfaßt, sind sie schwarz auf weiß wesentlich besser zu überschauen als auf dem Bildschirm. Der Bildschirm zeigt immer nur einen Ausschnitt. Auch Grafiken können mit dem Drucker ausgegeben werden. Einige schöne Beispiele sehen wir noch im Verlauf dieses Abschnitts.

Auf der Rückseite des Computers ist eine Steckerbuchse angebracht, die über ein spezielles Kabel mit dem Drucker verbunden wird. Der Apple-Drucker Imagewriter gehört zu den der Nadel- oder Matrixdruckern. Diese Drucker erzeugen ihre Zeichen und Grafiken mit mehreren übereinanderstehenden Nadeln.

Zur Ansteuerung des Druckers geben wir den Befehl

**PR#1**

ein. Diese Anweisung selektiert die Steckerverbindung 1, an der unser Drucker angeschlossen ist. Mit der Anweisung

**PR#0**

wird der Drucker in den Off-line-Status gesetzt. Jetzt kann wieder über die Tastatur eingegeben werden. Ist eines der Betriebssysteme geladen, muß der Drucker mit einem kleinen Programm aktiviert werden. Um z.B. ein BASIC-Listing auszudrucken, ist folgende Befehlssequenz notwendig:

**PR#1:LIST:PR#0**

Soll in einem BASIC-Programm ein Ausdruck erfolgen, ist vor die PRINT-Anweisung PR#1 zu setzen.

Beispiel:

```
10 PRINT "Das ist ein Druckertest"
20 PR#1
30 PRINT "Das ist ein Druckertest"
40 PR#0
50 PRINT "Ende des Druckertests"
60 END
```

Die erste PRINT-Anweisung wird auf dem Bildschirm ausgegeben. Danach schaltet Programmzeile 20 den Drucker an. Die folgende Ausgabeanweisung wird auf den Drucker geleitet. Erst nachdem PR#0 abgearbeitet ist, erscheint die Ausgabe wieder am Bildschirm. Nur die Programmzeile 30 wird auf dem Drucker ausgeführt.

Ausgabe am Bildschirm:

Das ist ein Druckertest
Ende des Druckertests

Ausgabe am Drucker:

Das ist ein Druckertest

In diesem Modus gibt der Drucker die Zeichen in Standardschrift aus. Wollen wir den ausgegebenen Satz z.B. unterstreichen, muß ein weiterer Code in das BASIC-Programm eingefügt werden. In unser erstes Beispiel fügen wir folgende Programmzeile ein:

**25 PRINT CHR$(27); "X"**

Damit wird der Druckercode für den Modus "Unterstreichen" eingestellt. Die PRINT-Anweisung überträgt den Code an den Drucker. Ergebnis:

<u>Das ist ein Druckertest</u>

```
PRINT CHR$(27);"Y"
```

Diese Anweisung schaltet den Unterstreichmodus aus. Alle anschließend ausgegebenen Daten werden wieder normal dargestellt. Es gibt aber noch eine Vielzahl weiterer Funktionen auf dem Drukker. Mit dem folgenden Programm stellt sich der Drucker mit seinen Möglichkeiten vor.

```
 10 PRINT "Das sind die Möglichkeiten des Apple Imagewriter"
 20 PRINT CHR$ (4); "PR#1"
 30 PRINT "Ich bin der Apple Imagewriter"
 40 PRINT CHR$(27);"X"
 50 PRINT "Das ist Unterstreichen"
 60 PRINT CHR$(27);"Y"
 70 PRINT CHR$(27);"!"
 80 PRINT "Das ist Fettdruck"
 90 PRINT CHR$(27);CHR$(34)
100 PRINT CHR$(14)
110 PRINT "Und das ist Breitschrift"
120 PRINT CHR$(15)
130 PRINT "Das ist wieder Normalschrift"
140 PRINT CHR$(4);"PR#0"
150 END
```

Der Ausdruck dazu:

Ich bin der Apple Imagewriter

Das ist Unterstreichen

**Das ist Fettdruck**

Und das ist Breitschrift

Das ist wieder Normalschrift

Mit dem Imagewriter können noch andere Druckmodi ausgeführt werden. Die Standardschriftsätze sind mit den Schaltern im Drucker eingestellt worden. Zu den Standardsätzen gehören US, British, German, French, Swedish, Italian, Spanish. Mit drei DIP-Schaltern lassen sich diese Schriftsätze aktivieren. Weitere Druckermodi zeigen die folgenden Ausdrucke.

Druckertest

unterstrichen

unterstrichen beendet

**Fettdruck**

Ende des Fettdrucks

Das ist der überschriftenmodus

Ende überschrift

Druckertest

unterstrichen

unterstrichen beendet

**Fettdruck**

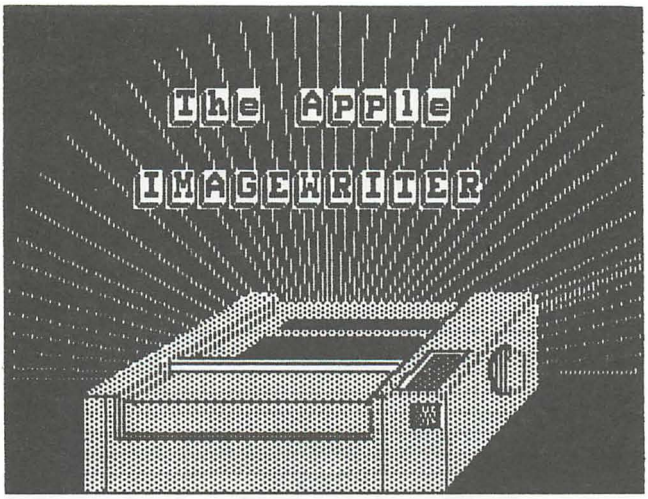
Ende des Fettdrucks

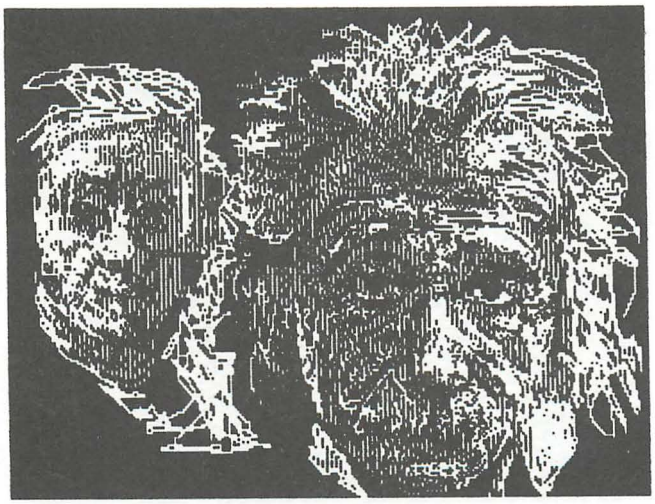
Das ist der überschriftenmodus

Ende überschrift

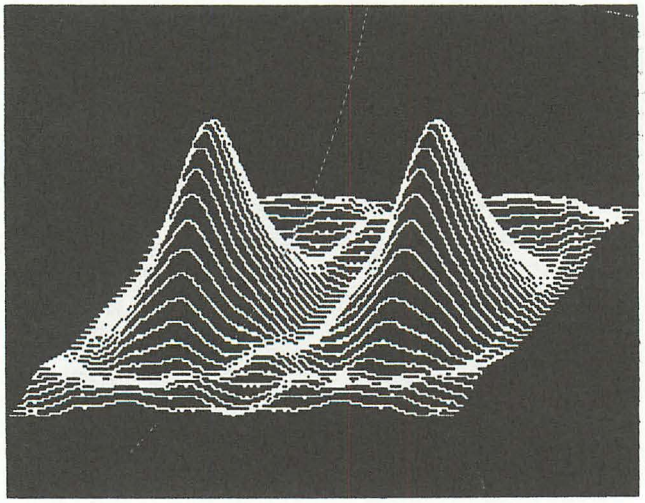
**Imagewriter Tool Kit**

Mit dem Imagewriter wird eine Diskette mitgeliefert, die das Arbeiten mit dem Drucker veranschaulichen soll. Sie haben u.a. auch die Möglichkeit, grafische Entwürfe auszudrucken. Die einzelnen Optionen finden Sie im Menü des Druckerprogramms.


The Apple
IMAGEWRITER



BECKMAN
BECKMAN



# 7

# Grafik mit Applesoft-BASIC

# 7 Grafik mit Applesoft-BASIC

Die große Herausforderung jedes BASIC-Programmierers ist die Erstellung von Grafiken und Diagrammen auf dem Computer. Dafür stellt uns der Apple IIc eine Menge guter Werkzeuge zur Verfügung. Anhand einfacher BASIC-Befehle ist es möglich, auch komplizierte Grafiken zu entwickeln.

## Aufruf der Grafikmodi

In den vorhergehenden Kapiteln wurde zur Darstellung der Texte und Zeichen die Textseite, die einen bestimmten Speicherplatz im Computer darstellt, benutzt. Ihr Apple IIc verfügt aber auch über Grafikseiten in verschiedenen Auflösungsstufen. Diese Grafikseiten sind Speicherblöcke, die im Arbeitsspeicher reserviert sind. Wir unterscheiden mehrere Grafikmodi:

- niedrige Auflösung mit Text (40 X 40 Bildpunkte)
- niedrige Auflösung Vollgrafik (40 X 48 Bildpunkte)

- niedrige Auflösung doppelt mit Text (80 X 40 Bildpunkte)
- niedrige Auflösung doppelt Vollgrafik (80 X 48 Bildpunkte)

- hochauflösende Grafik mit Text (280 X 160 Bildpunkte)
- hochauflösende Vollgrafik (280 X 192 Bildpunkte)

- doppelthochauflösende Grafik (560 X 160 Bildpunkte)
- doppelthochauflösende Vollgrafik (560 X 192 Bildpunkte)

## 7.1 Niedrigauflösende Grafik

Um einfache grafische Darstellungen zu programmieren, verfügt der Computer über eine niedrige Grafikauflösung. Es können in einer Bildzeile 40 Bildpunkte in Form kleiner Quadrate dargestellt werden. Die vertikale Auflösung hängt von der jeweils gewählten Darstellungsart ab.

**GR**

Mit der Anweisung GR schaltet der Computer vom Textmodus in die niedrige Grafikauflösung. Dabei werden vier Textzeilen am unteren Bildschirmrand reserviert, die für die Einflechtung von Texten in die Grafik gedacht sind. Wir können aber auch die unteren vier Textzeilen zusätzlich für die Grafik verwenden. Dazu geben wir ein:

POKE -16302,0

Mit diesem Befehl wird der Inhalt der Speicherstelle $C052 verändert. Es handelt sich bei dieser Speicherzelle um einen Soft-Switch (Software-Schalter). Der Computer benutzt ihn zur Statusanzeige.

Um aus der Vollgrafik niedriger Auflösung herauszukommen, müssen wir wieder den Soft-Switch umlegen. Nun können wir die vier Zeilen wieder für Texte verwenden. Dabei hilft uns der Befehl

POKE -16301,0

Der Speicherteil, der für die niedrige Grafikauflösung benutzt wird, ist der gleiche Speicherbereich wie für die Textdarstellung. Wenn Sie jetzt in den Textmodus umschalten, sehen Sie den Bildschirm voller Textzeichen, invers oder normal. Damit Sie wieder in den Textmodus kommen, benutzen Sie die Anweisung

**TEXT**

Sie kann auch wieder durch einen POKE-Befehl ersetzt werden. Dazu legen Sie den Soft-Switch der Speicherstelle dezimal -16303 um. Der Befehl dazu lautet

**POKE -16303,0**

Dieser Befehl kann nur dann richtig eingesetzt werden, wenn mit POKE -16304,0 in die niedrige Grafikauflösung gesprungen wurde.

Um die ersten Grafikzeichen programmieren zu können, müssen noch die Farben ausgewählt werden. Dazu verwenden Sie bitte den Befehl

**COLOR =**

Nach dem Ist-Gleich-Zeichen muß eine Zahl zwischen 0 und 15 folgen. Es werden 16 Farben zur Verfügung gestellt.

Tabelle 7-1: Farbcodes

| Codezahl | Farbe | Codezahl | Farbe |
|---|---|---|---|
| 0 | Schwarz | 8 | Braun |
| 1 | Fuchsienrot | 9 | Orange |
| 2 | Dunkelblau | 10 | Grau 2 |
| 3 | Purpur | 11 | Rosa |
| 4 | Dunkelgrün | 12 | Hellgrün |
| 5 | Grau 1 | 13 | Gelb |
| 6 | Blau | 14 | Aquamarinblau |
| 7 | Hellblau | 15 | Weiß |

Wenn Sie wollen, daß der folgende Grafikbefehl in blauer Farbe erscheint, müssen Sie die Anweisung COLOR=6 eingeben. Die Farben können, je nach Bildschirm, verschieden ausfallen. Bei monochromen Bildschirmen sind es verschiedene Graustufen in Form von Rastern.

### 7.1.1 Zeichnen mit PLOT

Der Computer stellt mit der Anweisung PLOT einen Bildpunkt in der mit COLOR gewählten Farbe dar. Die X- und Y-Koordinate des Bildpunktes wird, durch Komma getrennt, der Anweisung angehängt. Die erste Zahl gibt die Spaltennummer, von 0 bis 39, die zweite die Zeilen, von 0 bis 39, an. Bei Vollgrafik erhöht sich die Zeilennummer auf 47. Wenn die vorgeschriebene Spalten- oder Zeilenzahl überschritten wird, erscheint eine Fehlermeldung.

Beispiel:

```
10 GR
20 COLOR=15
30 PLOT 0,0:PLOT 39,0:PLOT 39,39:PLOT 0,39
40 END
```

Das kleine Programm stellt an allen vier Bildecken Bildpunkte in weißer Farbe dar. Die vier Zeilen Text werden von der Grafik dabei nicht benutzt. Um auch die restlichen 8 Grafikzeilen verwenden zu können, muß mit dem POKE-Befehl umgeschaltet werden. Fügen Sie folgende Programmzeilen zum vorangegangenen Programm:

```
40 POKE -16302,0
60 END
```

Der untere Teil der Grafikseite wird für die grafische Darstellung umgeschaltet. Die letzten vier Zeilen Text sind zu 8 Zeilen Grafik umgewandelt worden. Denn durch POKE -16302,0 wird die Grafik zugeschaltet, aber nicht gelöscht. Für die 39 x 39 Bildpunkte wird die Textausgabe unterdrückt. Die letzten vier Textzeilen müssen vom Basic-Programm gelöscht werden.

Programmänderung:

```
10 GR
20 POKE -16302,0
30 HOME : CALL -1998
40 COLOR=15
50 PLOT 0,0:PLOT 39,0:PLOT 39,39:PLOT 0,39
60 COLOR=2:PLOT 0,47:PLOT 39,47
70 END
```

Jetzt wird der Bildschirm mit CALL -1998 gelöscht. In Grafikzeile 47 sind die Eckpunkte mit dunkelblauen Bildpunkten markiert. Der Cursor blinkt in der Textzeile 22. Auf einem einfarbigen Bildschirm stellen sich die Bildpunkte farblich unterschiedlich dar.

Die Textzeichen sind im Grafikmodus aus zwei aufeinanderliegenden Bildpunkten zusammengesetzt. Das Byte, das den hexadezimalen Wert des Textzeichens darstellt, wird in zwei Teile zerlegt. Bit 7 bis 4 bestimmt die Farbe des oberen Bildpunktes. Bit 3 bis Bit 0

stellt den Farbcode des unteren Bildpunktes dar. Der Wert $A0 zeigt an, daß der obere Bildpunkt schwarz, der untere grau 2 gefärbt ist.

Tabelle 7-2: Speicheraufbau der Textseite 1

| | Spalte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | ..... | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Hex $ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 2 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | ..... | 7 |
| Hex | Dez | | | | | | | | | | | | | | | | |
| $400 | 1024 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | | |
| $480 | 1152 | * | * | * | * | * | * | * | * | * | | | | | | | |
| $500 | 1280 | * | * | * | * | * | | | | | | | | | | | |
| $580 | 1408 | * | * | * | * | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| $6D0 | 1744 | | | | | | | | | | | | | | | | |
| $750 | 1872 | | | | | | | | | | | | | | | | |
| $7D0 | 2000 | | | | | | | | | | | | | | | | |

### 7.1.2 Bildpunkte ermitteln mit SCRN

Die Anweisung SCRN kann den Farbwert eines Bildpunktes analysieren. Es muß dazu die X/Y-Koordinate angegeben werden. Aus der Position und dem hexadezimalen Wert ermittelt diese Funktion den dezimalen Wert des Farbcodes.

Beispiel:

```
10 GR
20 COLOR=11
30 PLOT 2,2
40 A=SCRN(2,2)
50 PRINT "A=";A
```

In Zeile 40 wird der Wert, den der Bildpunkt 2,2 besitzt, errechnet und der Variablen A zugeordnet. Programmzeile 50 zeigt den Wert am Bildschirm an.

### 7.1.3 Horizontale Linien zeichnen mit HLIN

Im niedrigauflösenden Grafikmodus gibt es die Möglichkeit, Linien mit einer kurzen Befehlssequenz aufzurufen. Dabei wird von einem Knotenpunkt aus die Linie gezogen. Für HLIN wird der Ausgangspunkt mit Spaltenanfangs- und -endnummer festgelegt. Dabei ist die erste Zahl die Anfangs-, die zweite Zahl die Endspaltennummer.

Der Befehlszusatz AT sagt aus, daß die Linie Spalte X bis Spalte Y in Zeile Z gesetzt wird. Der Endwert zeigt die Zeilennummer an.

Beispiel:

```
10 GR
20 COLOR=2
30 HLIN  0,39 AT 5
40 HLIN  0,39 AT 10
50 HLIN  10,30 AT 15
60 HLIN  0,39 AT 39
70 END
```

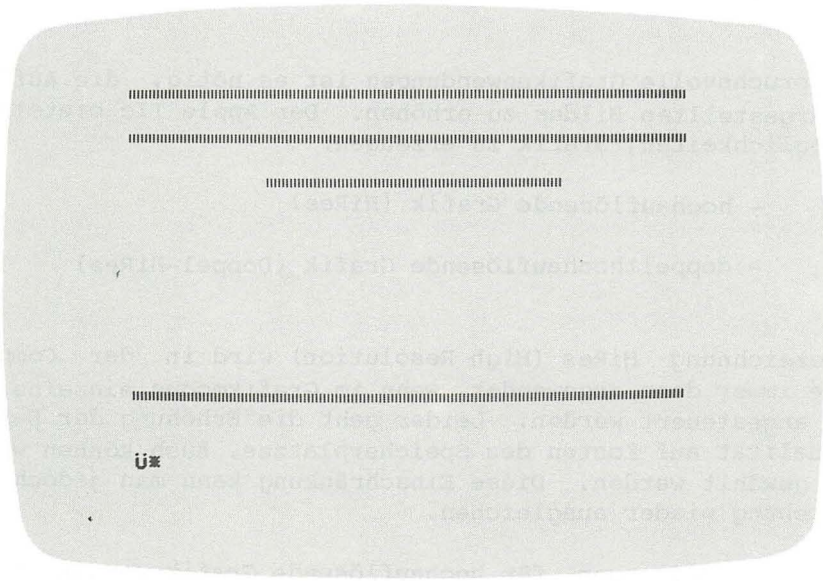



Abb. 7-1: HLIN

### 7.1.4 Vertikale Linien mit VLIN

Vertikale Linien werden mit dem Befehl VLIN gezogen. Dabei geben die ersten beiden Zahlen die Zeilennummer an. Die letzte Zahl gibt die Spaltennummer an. Die Anweisung

```
VLIN 0,39 AT 10
```

zieht eine vertikale Linie von Zeile 0 bis Zeile 39 in der Spalte 10.

## 7.2 Hochauflösende Grafik

Für anspruchsvolle Grafikanwendungen ist es nötig, die Auflösung des dargestellten Bildes zu erhöhen. Der Apple IIc bietet dazu zwei Möglichkeiten, Grafik zu erzeugen:

- hochauflösende Grafik (HiRes)
- doppelthochauflösende Grafik (Doppel-HiRes)

Die Bezeichnung HiRes (High Resolution) wird in der Computersprache immer dann angewendet, wenn im Grafikmodus einzelne Bildpunkte angesteuert werden. Leider geht die Erhöhung der Darstellungsqualität auf Kosten des Speicherplatzes. Auch können weniger Farben gewählt werden. Diese Einschränkung kann man jedoch durch Farbmischung wieder ausgleichen.

Die BASIC-Anweisungen für hochauflösende Grafik werden nur in Applesoft-BASIC zur Verfügung gestellt. Dennoch ist es möglich, durch gezielte Systembefehle auch in Integer-BASIC hochauflösende Grafik zu programmieren.

### 7.2.1 HiRes-Grafikseite 1

Wie schon bei der niedrigen Grafikauflösung werden bei der HiRes-Grafikseite 1 vier Zeilen Text auf dem Bildschirm dargestellt. Die Auflösung in diesem Modus beträgt 280 x 160 Bildpunkte. Sollen 560 Bildpunkte horizontal erscheinen, muß die 80-Zeichen-Karte eingeschaltet werden. Dabei verschieben sich die Grafikbereiche. Der Speicherbereich $2000 bis $3FFF überlagert sich, denn die Adressen der 64 KByte sind zweimal vorhanden. So werden insgesamt 32 KByte für die 560 x 192 Bildpunkte reserviert.

**HGR**

Mit dieser Anweisung schaltet der Rechner auf die HiRes-Seite 1 um. Alle Bildpunkte, die gesetzt waren, sind zugleich gelöscht worden. Das Umschalten kann auch mit einem POKE-Befehl erfolgen. Dabei wird nur umgeschaltet, aber nicht gelöscht. Die gesetzten Bildpunkte bleiben unverändert stehen.

Beispiel:

```
10 HGR
```

(Der Rechner schaltet auf die HiRes-Seite 1 um und löscht den Bildschirm)

```
10 POKE -16304,0
```

(Es wird auf den Grafikmodus geschaltet)

```
20 POKE -16297,0
```

(schaltet auf die hochauflösende Grafik)

```
30 POKE -16300,0
```

(schaltet die Grafikseite 1 ein)

Wie schon bei der niedrigen Grafikauflösung, so kann auch hier der Textteil abgeschaltet werden. Dazu verwenden Sie bitte

```
POKE -16302,0
```

Die Darstellung beträgt jetzt 280 x 192 Bildpunkte. Das sind 53760 Bildpunkte, die einzeln angesteuert werden.

### 7.2.2 Farben setzen mit HCOLOR

Auch diese Anweisung kennen wir in abgeänderter Form schon von der niedrigen Grafikauflösung. Damit ist es möglich, 7 Farben anzuwählen. Schwarz und Weiß zählt als Farbe, Weiß ist in zwei Tönen vertreten.

Tabelle 7-3: Farbcode für hochauflösende Grafik

| Code | Farbe |
|---|---|
| 0 | Schwarz |
| 1 | Grün |
| 2 | Violett |
| 3 | Weiß |
| 4 | Schwarz |
| 5 | Orange |
| 6 | Blau |
| 7 | Weiß |

Soll der nächste ausgegebene Zeichenbefehl in Grün dargestellt werden, so benutzen wir den Befehl HCOLOR = 1. Die Farbe bleibt so lange im Pufferbereich stehen, bis eine neue gewählt wird. In der Speicherzelle dezimal 28 ($1C) steht der Farbcode, der mit HCOLOR definiert wurde. Sie ist dann wichtig, wenn zwischen einem HPLOT-Befehl die Farbe geändert wird.

### 7.2.3 HPLOT

Die Anweisung kann einen Punkt an einer X/Y-Koordinate auf dem Bildschirm in der Farbe zeichnen, die vorher mit HCOLOR definiert wurde. Im hochauflösenden Grafikmodus können aber auch Linien in jeder Richtung gezeichnet werden.

Beispiel:

```
10 HGR
20 HCOLOR = 3
30 HPLOT 10,20
```

Das Beispiel zeichnet einen weißen Punkt in der Zeile 20, Spalte 10. Zeile 10 und 20 im Programm geben den Grafikbereich und die Farbe an, in der geplottet wird. Soll eine Linie vom Punkt 20,20 zu dem Koordinatenpunkt 50,50 gezeichnet werden, so ist folgende Befehlsfolge einzugeben:

```
HPLOT 20,20 TO 50,50
```

Da der Grafikbereich und die Farbe schon gesetzt sind, ist nur noch der Plot-Befehl einzugeben. Dies ist auch in der Betriebsart Direktanweisung möglich. Dabei ist sofort das Ergebnis auf dem Bildschirm sichtbar. Soll eine weitere Linie an die erste anschließen, ist der Befehl

**HPLOT TO**

anzufügen. Die Linie wird bei unserem Beispiel ab dem Koordinatenpunkt 50,50 gezeichnet.

Beispiel:

```
HPLOT TO 100,50
```

Die Linie wird bis zum Punkt 100,50 in der Farbe Weiß gezogen. Fügen wir vor den HPLOT-TO-Befehl eine HCOLOR-Anweisung ein, müßte die nächste HPLOT-Anweisung in einer neuen Farbe ausgeführt werden.

Beispiel:

```
10 HGR
20 HCOLOR = 3
30 HPLOT 20,20 TO 100,100
40 HCOLOR = 2
50 HPLOT TO 190,100
60 HPLOT TO 110,20
```

Wie Sie aber sehen können, sind die Linien, die ihre Richtung geändert haben, noch immer in der Farbe Weiß angezeigt. Denn bei einer HPLOT-TO-Anweisung, wird immer die zuletzt definierte Farbe des letzten HPLOT-TO-Befehls verwendet.

```
70 HPLOT 110,20 TO 20,20
```

Wird diese Programmzeile angehängt, erscheint die vierte Linie in violetter Farbe. Um aber schon bei der Zeile 50 violett zeichnen zu können, muß die Speicherzelle 28 mit POKE umgeschaltet werden. Ändern Sie Zeile 40 wie folgt ab:

```
40 HCOLOR = 2: POKE 28,85
```

Jetzt sind alle drei Linien mit der richtigen Farbe abgebildet. Um die anderen Farbcodes ausfindig zu machen, muß nur der Befehl PRINT PEEK(228) verwendet werden. Das ist die Speicherzelle mit dem zuletzt eingegebenen Farbcode.

Werden die nächsten Programmzeilen an das letzte Beispiel angehängt, entsteht beim Starten mit RUN ein Effekt, der für eine spätere Manipulation der Grafikseiten benutzt wird.

```
80 HGR 2
90 HCOLOR = 2
100 HPLOT 100,100 TO 190,100
110 HPLOT TO 190,10
120 HPLOT TO 100,10
130 HPLOT TO 100,100
140 GOTO 10
```

Die hinzugefügten Zeilen bewirken ein Umschalten zwischen den zwei Grafikseiten. Dabei werden die Bilder immer wieder neu aufgebaut. Die Anweisungen HGR und HGR2 löschen ihren jeweiligen Grafikbereich. Mit den Anweisungen

```
POKE -16299,0
```

und

```
POKE -16300,0
```

kann man diesen Effekt wesentlich beschleunigen. Ändern Sie die Programmzeilen wie folgt ab:

```
140 POKE -16299,0
150 POKE -16300,0
160 GOTO 140
```

Die POKE-Anweisungen in Zeile 140 und 150 schalten zwischen der Grafikseite 1 und 2 um. Mit dieser Art der Umschaltung können bewegte Grafiken erstellt werden. Wir sprechen dann von einer Animation. Die Bilder werden auf den Grafikseiten erstellt und dann übereinandergeblendet. Mit Hilfe des Softschalters entsteht so eine Bewegung. Das Bild, das dem dargestellten Bild folgen soll, muß in seiner Form oder Lage vorher aber verändert werden.

Wollen Sie Ihr Listing wieder anschauen, muß die Anweisung

**TEXT**

eingegeben werden. Dabei springt der BASIC-Interpreter von der Grafikseite in den Textmodus. Haben Sie jetzt eine Grafik auf dem Bildschirm erstellt und wollen diese wieder anschauen, so könnten Sie mit HGR oder HGR2 in die jeweilige Grafikseite gehen. Doch diese Anweisungen löschen beim Zugriff den Bildschirm. Um dennoch die Grafiken anzuschauen, gibt es die Softschalter. Mit ihnen kann vom Textmodus in den Grafikmodus geschaltet werden, ohne den Speicherbereich der jeweiligen Grafikseite zu löschen.

**POKE -16304,0**

Dieser Befehl schaltet vom Textmodus in den Grafikmodus. Dabei ist entscheidend, welcher Grafikmodus gewählt wurde.

**POKE -16302,0**

Mit dieser POKE-Anweisung wird der Schalter betätigt, der die Vollgrafik einschaltet. Die Textzeilen werden durch Grafikteile ersetzt.

**POKE -16301,0**

schaltet die vier Zeilen Text ein. Die Grafikseite ist jetzt unterteilt in Grafik und Text.

**POKE -16300,0** schaltet die Grafikseite 1
**POKE -16299,0** schaltet die Grafikseite 2
**POKE -16298,0** schaltet die niedrigauflösende Grafik
**POKE -16297,0** schaltet die hochauflösende Grafik

## 7.3 Speichern von Grafikseiten

Die Grafikseiten, die Sie erstellt haben, sind oft mühsam programmiert worden. Um dennoch die Grafiken schnell zeigen zu können, müssen die Grafikseiten auf Diskette gesichert werden. Die Grafikseiten selbst stellen einen Teil des Arbeitsspeichers dar, sie müssen also nur noch die Daten abspeichern. Wir wollen dies mit dem DOS-Befehl **BLOAD** tun. Die Wirkungsweise wird im Abschnitt über den Diskettenbetrieb (s. Kap 9 und 10) genauer beschrieben. Für uns ist jetzt nur der Befehlsablauf wichtig.

**BSAVE Dateiname A$2000,L$2000** speichert die Grafikseite 1

**BSAVE Dateiname A$4000,L$2000** speichert die Grafikseite 2

Für das Wort Dateiname bezeichnen wir unsere Grafik. Soll die gespeicherte Grafik wieder in den Computer geladen werden, so muß statt dem Befehl BSAVE der Befehl BLOAD eingegeben werden. Speicherplatz und Speicherumfang, die bei BSAVE angegeben wurden, können bei BLOAD weggelassen werden.

## 7.4 Bilder-Animation

Für Computerspiele und bewegte Grafiken wird das Übereinanderlegen von Bildsegmenten zur Täuschung des menschlichen Auges benutzt. In der einfachsten Form können wir das bereits mit Textzeichen erreichen. Das folgende Programm läßt einen Stern über den Bildschirm wandern. Dabei wird die Position des Sterns mit HTAB angegeben. Die Position wird dabei wechselseitig beschrieben oder durch ein Leerzeichen gelöscht.

Einfaches Beispiel:

```
10  HOME : VTAB 12
20  FOR I = 1 TO 39
30  HTAB I: PRINT "*";
40  FOR N = 1 TO 50: NEXT N
50  HTAB I: PRINT " ";
60  NEXT I
70  GOTO 10
```

Mit dem Abarbeiten der FOR-NEXT-Schleife in der Programmzeile 40 kann der Bewegungsablauf gesteuert werden. Je kleiner die größte Zahl N ist, desto schneller wandert der Stern über den Bildschirm. Das nächste Beispiel zeigt eine bewegte Animation in Form eines Quadrats, das in einem Rechteck umherwandert. Geben Sie das Programm sorgfältig ein, und starten Sie es anschließend.

```
10 SL = 6 * 4096:SC = 8:X = 100:Y = 50:DX = 1:DY = 1
20  FOR I = SL TO SL + 6: READ A: POKE I,A: NEXT
30  POKE 232,0: POKE 233,96
40  HGR : POKE  - 16302,0: HCOLOR= 3: ROT= 0
50  HPLOT 0,0 TO 250,0 TO 250,152 TO 0,152 TO 0,0
60  SCALE= SC: XDRAW 1 AT X,Y
70 OX = X:OY = Y:X = X + DX:Y = Y + DY
80  IF X > 248 - SC OR X < 2 THEN DX =  - DX:SF = 1
90  IF Y > 150 - SC OR Y < 2 THEN DY =  - DY:SF = 2
100  XDRAW 1 AT OX,OY: XDRAW 1 AT X,Y
110  GOTO 70
120  DATA   1,0,4,0,53,39,0
```

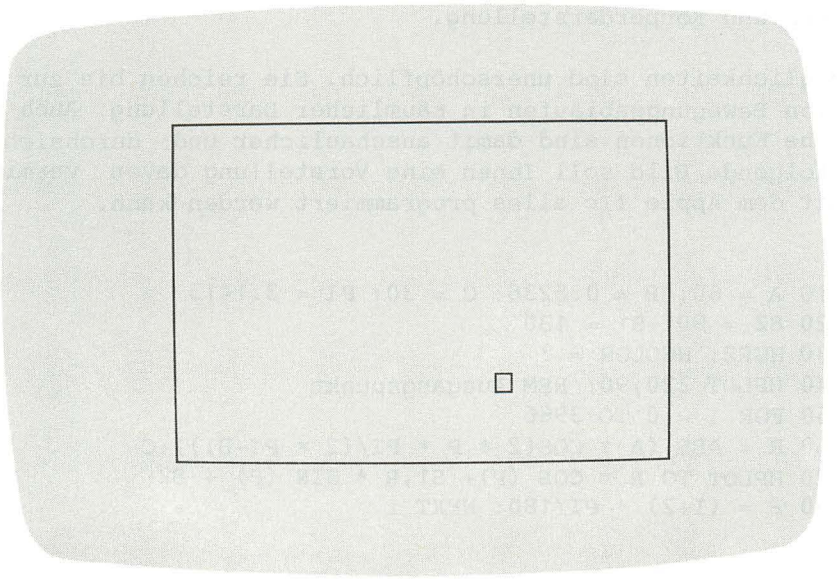


Abb. 7-2: Ein Animation

## 7.5 3D-Darstellungen

Die ausführliche Beschreibung der Möglichkeiten der 3D-Darstellung ist im Rahmen dieses Buches nicht möglich. Für alle, die sich mit dieser Materie beschäftigen wollen, sei auf entsprechende Fachliteratur hingewiesen. In diesem Abschnitt möchte ich Ihnen nur einige Anregungen und Tips geben, die es Ihnen ermöglichen, Grafiken in räumlicher Darstellung zu erzeugen.

Wie schon unser letztes Beispiel gezeigt hat, kann mit einfachsten Mitteln bereits ein räumlicher Effekt entstehen. Da der Computer nur ein zweidimensionales Bild darstellen kann, müssen wir dem Betrachter eine räumliche Tiefe des Bildes vortäuschen. Mit perspektivischen Darstellungen ist es möglich, räumliche Tiefe und Dreidimensionalität zu erzeugen. Dabei sind Objekte im Vordergrund größer gezeichnet als die im Hintergrund.

Sollen aufwendige Darstellungen erzeugt werden, kommen Sie an einer mathematischen Berechnung nicht mehr vorbei. Um räumliche Objekte berechnen zu können, sind verschiedene Winkel und Vektoren zu berücksichtigen. Es wird unterschieden zwischen Gitterdarstellung und Körperdarstellung.

Die Möglichkeiten sind unerschöpflich. Sie reichen bis zur Animation von Bewegungsabläufen in räumlicher Darstellung. Auch mathematische Funktionen sind damit anschaulicher und durchsichtiger. Das folgende Bild soll Ihnen eine Vorstellung davon vermitteln, was mit dem Apple IIc alles programmiert werden kann.

```
10 A = 60: B = 0.5236: C = 30: PI = 3.1415
20 S2 = 90: S1 = 130
30 HGR2: HCOLOR = 3
40 HPLOT 220,90: REM Ausgangspunkt
50 FOR I = 0 TO 3966
60 R = ABS (A * COS(2 * P * PI/(2 * PI-B)))+C
70 HPLOT TO R * COS (P)+ S1,R * SIN (P) + S2
80 P = (I+2) * PI/180: NEXT I
```

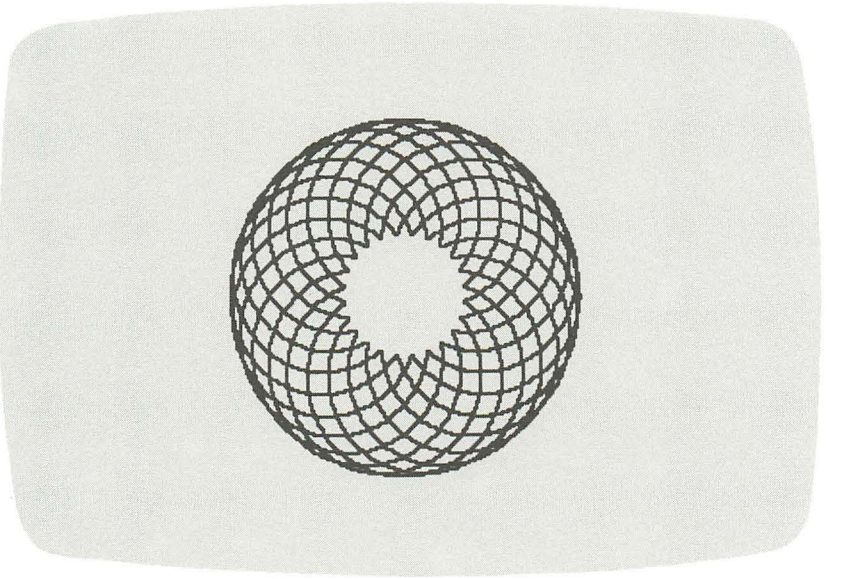


Abb. 7-3: PLOT auf dem Bildschirm

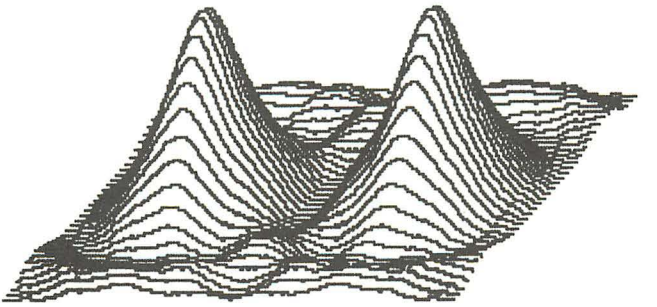


Abb. 7-4: Mathematische Funktion in 3D-Darstellung

# 8

# Das Betriebssystem DOS 3.3

# 8 Das Betriebssystem DOS 3.3

Der Arbeitsspeicher des Apple IIc würde bei weitem nicht ausreichen, alle Programme und Daten darin zu speichern. Außerdem gingen diese beim Ausschalten des Rechners verloren.

Um die Daten abspeichern zu können, bedarf es eines Datenträgers, der in der Lage ist, Daten aufzunehmen, wieder abzugeben und zu löschen. Die schnellste und billigste Art ist derzeit noch das Abspeichern auf Diskette.

Der Apple IIc besitzt in seiner Grundversion bereits ein eingebautes Diskettenlaufwerk. Ein Anschluß für ein Kassettenlaufwerk ist serienmäßig nicht vorgesehen. Die Abspeicherung der Daten wird von einem übergeordneten Programm im Speicher des Computers überwacht. Das Programm hat den Namen DOS, das ist die Abkürzung von "Disk Operating System", zu deutsch Diskettenverwaltungsprogramm. Wir wollen uns in diesem Kapitel ausschließlich mit der Speicherung und Verwaltung von Daten auf Diskette befassen.

## 8.1 Die Diskette

Als Diskette bezeichnen wir einen Datenträger in Form einer geschützten Kunststoffscheibe, die (beidseitig) mit einer magnetisierbaren Schicht versehen ist. Die Diskette wird auch als Disk oder Floppydisk bezeichnet. Es gibt eine Vielzahl von Diskettentypen. Der Apple IIc benutzt die 5 1/4-Zoll-Diskette. Darüber hinaus werden aber auch 8 Zoll, 3 Zoll, 3 1/5 Zoll angeboten. Das sogenannte Zentrierloch, das bei qualitativ guten Disketten mit einem Verstärkungsring versehen sein sollte, zentriert die Diskette im Laufwerk. Die Lebensdauer und Laufsicherheit einer Diskette wird mit einem Verstärkungsring wesentlich erhöht.

Beim Formatieren wird die Diskette in definierte Bereiche gegliedert. Sie ist in 16 Sektoren und 35 Spuren aufgeteilt. Abbildung 8-2 zeigt die Aufteilung.

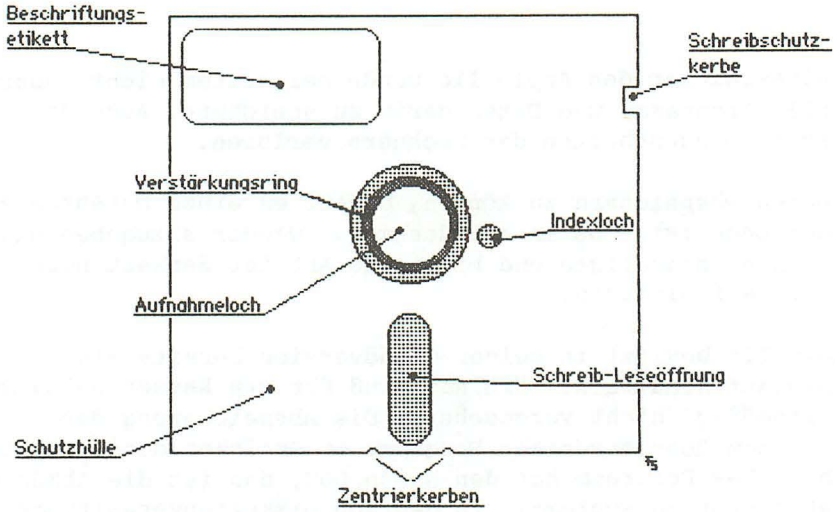



Abb. 8-1: Eine 5 1/4-Zoll-Diskette

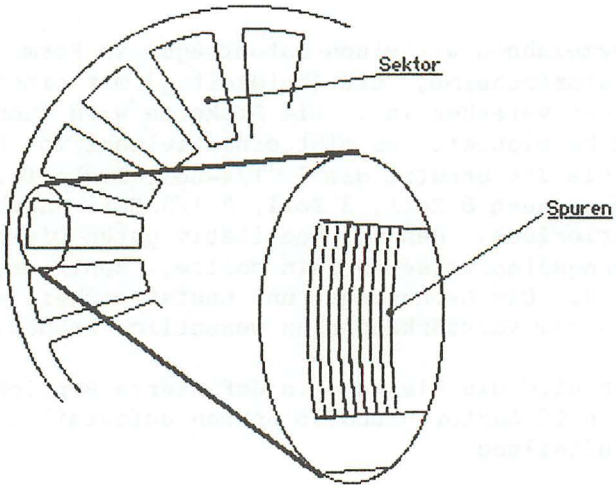



Abb. 8-2: Spuren und Sektoren

## 8.2 Der Schreibschutz

Die Diskette kann mechanisch vor Überschreiben geschützt werden. Dazu dient die Einkerbung an der linken Seite der Diskette. Durch Überkleben mit einem Klebestreifen wird die Diskette geschützt. Ein kleiner Schalter, der im Diskettenlaufwerk eingebaut ist, tastet beim Einschieben der Diskette die Schutzkerbe ab. Ist diese Überklebt, wird ein Beschreiben der Diskette verhindert. Ist kein Klebestreifen aufgebracht, so kann auf die Diskette geschrieben werden.

## 8.3 Umgang mit Disketten

Oft wird die Diskette auch als "Floppydisk" bezeichnet. Die Flxibilität hat jedoch ihre Grenzen. Auch wenn die Diskette flexibel ist, sollte sie dennoch nicht gebogen werden. Eine mechanische Beschädigung wäre die Folge.

Halten Sie Ihre Disketten von jeder magnetischen Strahlung fern. Dies sind zum Beispiel Fersehgeräte, Monitore, Lautsprecher und Elektrogeräte. Die magnetischen Felder, die diese Geräte ausstrahlen, verändern die Lage der magnetischen Aufzeichnungen auf Ihren Disketten. Dabei kann nicht nur die Information der Diskette verlorengehen, auch eine Wiederverwendung kann unter Umständen nicht mehr möglich sein.

Sie sollten die Disketten nie mit einem Bleistift beschriften. Die Mienenpartikel, die dabei entstehen, können in das Schreib-/Lesefenster der Diskette fallen und deren Oberfläche bzw. den Tonkopf beschädigen. Verwenden Sie zum Beschriften Aufklebeetiketten, die jeder Diskettenverpackung beiliegen.

Behandeln Sie die Disketten sorgfältig. Beim Einschieben in das Laufwerk sollte mit Gefühl vorgegangen werden. Fassen Sie niemals in das Schreib-/Lesefenster der Diskette. Fingerabdrücke können die Diskette unbrauchbar machen. Auch sollte die Diskette keiner Wärmeeinstrahlung ausgesetzt werden.

Wenn Sie diese Anweisungen beachten, können Sie die Lebensdauer Ihrer Diskette wesentlich erhöhen und werden kaum Lesefehler ver-

zeichnen. Wenn Sie bedenken, daß Datenträger in der Groß-EDV in klimatisierten Räumen gelagert werden, so sollten Sie Ihren Disketten eine entsprechend sorgfältige Behandlung zukommen lassen.

## 8.4 DOS 3.3

Das Betriebssystem DOS 3.3 hat die Aufgabe, den Austausch von Daten zwischen Computer und Diskettenlaufwerk zu verwalten. Auf jeder Diskette, die das DOS-3.3-Format besitzt, ist das komplette Betriebssystem gespeichert. Wird die Diskette gestartet, lädt sich DOS automatisch in den dafür vorgesehenen Speicherbereich. Die Diskette kann mit folgendem Befehl "gebootet" (gestartet) werden:

**PR#6**

Der Steckplatz 6 wird angesprochen. An ihn ist der Diskettencontroller angeschlossen. Wird der Apple IIc eingeschaltet, läuft die Laderoutine automatisch ab. Steckt eine DOS-3.3-Diskette im Laufwerkschacht, so wird beim Einschalten DOS geladen. Alle weiteren Diskettenbefehle sind jetzt von DOS abhängig. Nur DOS-formatierte Disketten sind natürlich unter DOS 3.3 lauffähig.

### 8.4.1 Initialisieren mit INIT

Mit INIT wird die Diskette in sog. Sektoren und Spuren (tracks) gegliedert. Die Strukturierung dient bei der Speicherung von Daten der genauen "Platzanweisung". Eventuell auftretende Fehler können somit schneller lokalisiert werden. Da an den Apple IIc zwei Diskettenlaufwerke anschließbar sind, sind diese auch zu selektieren. Ist DOS geladen, und der Befehl INIT soll angewendet werden, so muß sich im BASIC-Speicher ein Programm befinden. Ansonsten erscheint keine Meldung am Bidschirm außer dem Cursor.

Beispiel:

```
10 PRINT "DOS 3.3"
20 PRINT "Franz Santjohanser"
30 PRINT "angelegt am 16.10.2001"
```

Jetzt können wir unseren Befehl INIT starten. Wir wollen es HELLO nennen. Der Befehl lautet also vollständig:

**INIT HELLO**

Die Befehle für DOS müssen immer in GROSSBUCHSTABEN eingegeben werden. An die Anweisung INIT kann z.B. noch das Diskettenlaufwerk angehängt sein. Dies ist auch bei den anderen DOS-Befehlen möglich.

Beispiel:

**INIT HELLO, D2**

Jetzt wird das externe Laufwerk angesprochen. Die Diskette wird in Spuren und Sektoren eingeteilt. Sie kann jetzt als neue DOS-Diskette verwendet werden. Legen Sie die initalisierte Diskette in das Laufwerk, und starten Sie mit PR#6 die Diskette neu. Auf dem Bildschirm erscheint der Text, der als kleines Programm eingegeben wurde. Denn DOS startet das HELLO-Programm. Überprüfen Sie das mit der Anweisung LIST.

Der Diskette kann eine individuelle Volumennummer zugeordnet werden, die auf die Diskette gespeichert wird. Sie dient zur späteren Markierung der einzelnen Disketten. Dazu gibt man hinter dem Programmnamen, nach dem Komma, eine Zahl zwischen 0 und 255 an.

Beispiel:

**INIT HELLO, D1, V178**

Es wird im Diskettenlaufwerk 1 die Diskette mit der Volumennummer 178 initialisiert. Um zu sehen, was sich auf der Diskette befindet, geben wir den Befehl

CATALOG

ein. Damit wird das Directory (Inhaltsverzeichnis) der Diskette ausgegeben. Sind mehr Programmnamen auf der Diskette enthalten als auf den Bildschirm passen, erscheint zuerst ein Teil und auf den Tastendruck einer beliebigen Taste der Rest. In unserem Fall ist die Ausgabe:

**DISK VOLUME 178**

**A 002 HELLO**

Sie sehen, daß die Diskette die Volumennummer bekommen hat. Das A vor 002 bedeutet, daß die Daten, die abgelegt wurden, in Applesoft-BASIC geschrieben sind. Die 002 sagt aus, daß das HELLO-Programm zwei Blöcke der Diskette belegt. Ein Block faßt maximal 256 Byte. Ist ein Programm größer als 255 Blöcke, dann beginnt die Zahl wieder bei 000. Wollen Sie ein Programm speichern, dann muß ein korrekter Dateiname angegeben werden. Der Programmname ist maximal 30 Zeichen lang. Dabei dürfen alle Zeichen außer dem Komma benutzt werden. Längere Namen sind möglich, aber dann auf 30 Zeichen gekürzt.

Beispiel:

TEXT
GRAFIK
MUSIK usw.

Die Datei- oder Programmnamen sollten in Zusammenhang mit dem jeweiligen Inhalt stehen.

### 8.4.2 RENAME

Programmnamen können mit dieser Anweisung umbenannt werden. Der alte Programmname wird hinter RENAME gestellt. Dann folgt, nach einem Komma, der neue Programmname. Im folgenden Beispiel soll das Programm FRANZ in OTTO umbenannt werden.

Beispiel:

```
RENAME FRANZ, OTTO
```

DOS erlaubt bei der Eingabe Leerzeichen. Sie werden aber bei der Abarbeitung der Anweisung nicht berücksichtigt.

Beispiel:

**C A TAL OG**

Der Befehl, der das Inhaltsverzeichnis anzeigt, wird ohne Syntaxfehlermeldung ausgeführt. Soll ein Programmname oder eine Datei gelöscht werden, muß die Anweisung

**DELETE**

gegeben werden. Hinter DELETE folgt der Programm- oder Dateiname. Wird das Programm auf der Diskette von DOS nicht gefunden, folgt die Fehlermeldung:

**FILE NOT FOUND**

Sie sollten das Programm, das Sie löschen wollen, genau anschauen und sorgfältig eingeben. Der Befehl kann nicht rückgängig gemacht werden.

**SAVE**

Der Befehl SAVE (retten) speichert das Programm ab, das sich im BASIC-Arbeitsspeicher befindet. Um es mit dem Namen RECHNEN abspeichern zu können, ist folgende Befehlssequenz einzugeben:

**SAVE RECHNEN**

Ist das Programm länger als die restliche Speicherkapazität auf der Diskette, gibt DOS folgende Fehlermeldung aus:

**DISK FULL**

Sie können eine neue Diskette formatieren und dann das Programm auf dieser Diskette abspeichern.

**LOAD**

Die Anweisung LOAD benutzen wir zum Einlesen von Programmen, die sich auf Diskette befinden. Das Programm RECHNEN aus dem vorherigen Beispiel wird folgendermaßen geladen:

**LOAD RECHNEN**

Sie müssen das Programm mit RUN starten. Auch diese Anweisung können Sie umgehen. Geben Sie dazu nicht LOAD RECHNEN ein, sondern **RUN RECHNEN.** Das Programm wird eingeladen und von DOS gestartet. Programme, die nicht gelöscht werden sollen, kann man dagegen schützen. Dazu gibt es die Anweisung

**LOCK**

Nach LOCK muß der Programmname angegeben werden, der geschützt werden soll. Mit der Anweisung **UNLOCK** wird dieser Schutz wieder aufgehoben. Im Inhaltsverzeichnis der Diskette sind die geschützten Programme mit einem Stern gekennzeichnet.

**VERIFY**

Diese Anweisung überprüft den Inhalt des Arbeitsspeichers mit dem abgespeicherten Programm auf Diskette. Dabei werden die Informationen verglichen. Besonders bei defekten und verschmutzten Disketten ist diese Überprüfung notwendig. Wird ein Fehler festgestellt, erscheint die Fehlermeldung

**I/O ERROR**

Dabei überprüft VERIFY nur die technischen Informationen, nicht aber den Inhalt auf Richtigkeit. Ein fehlerhaftes BASIC-Programm wird damit nicht entdeckt. Mit VERIFY können alle Datenformate, die in DOS gespeichert wurden, mit dem Speicher des Computers verglichen werden. Die Voraussetzung ist, daß die Daten auch tatsächlich im Arbeitsspeicher vorhanden sind. Wird ein Programmname aufgerufen, der sich nicht auf der Diskette befindet, erfolgt die Fehlermeldung

**FILE NOT FOUND**

VERIFY ist von Applesoft-BASIC und Integer-BASIC aus verwendbar. Auch im Monitor ist VERFIY aktiv.

## MON und NOMON

Mit der Anweisung ist es möglich, Ein- und Ausgabefunktionen der Diskette zu überwachen. MON stellt dabei sieben verschiedene Möglichkeiten zur Verfügung. Hinter MON muß einer der Buchstaben C, I oder O stehen. C steht für Befehle, die an die Diskette gegeben werden. I signalisiert eine Eingabe, die von der Diskette erfolgt. O ist der Anfangsbuchstabe von Output und bedeutet, daß die Übertragung zur Diskette überwacht wird.

| | |
|---|---|
| MON C | Befehle an die Diskette |
| MON I | Eingaben von der Diskette |
| MON O | Ausgabe an die Diskette |
| MON I,O | Ein- und Ausgabe |
| MON C,I | Befehle und Eingabe |
| MON C,O | Befehle und Ausgabe |
| MON C,I,O | Befehle, Ein- und Ausgabe |

Es muß mindestens ein Parameter angegeben werden. NOMON mit den entsprechenden Parametern schaltet die einzelnen Funktionen wieder ab. Die Buchstaben sind durch Komma zu trennen.

## 8.5 DOS-Befehle in Programmen

In BASIC-Programmen müssen Daten von der Diskette geladen oder auf Diskette gespeichert werden. Dabei sind die DOS-Befehle etwas anders zu handhaben als in der Betriebsart Direktanweisung. Um einen DOS-Befehl in einem Programm ausführen zu können, muß dieser als Ausdruck eines Ausgabecodes definiert sein.

Beispiel:

```
10 CHR$(4); "LOAD RECHNEN"
```

Die Befehlssequenz CHR$(4) erzeugt den Code für CONTROL-D. Die Anweisung muß in Anführungszeichen stehen. Es kann auch der Tastencode CTRL-D direkt eingegeben werden. Dazu definieren wir eine Variable mit dem Inhalt CTRL-D.

Beispiel:

```
10 A$ = "": REM CTRL-D
20 A$;"CATALOG"
```

In den Anführungszeichen wurde die Tastenfolge CONTROL-Taste und die Taste D gedrückt. Der REM-Zusatz sollte mit angegeben sein.

### 8.5.1 Sequentielle Textdateien

Programme in BASIC und Maschinensprache sind leicht abzuspeichern oder zu laden. Bei Textdaten ist das nicht ganz so einfach. Um diese Daten abzuspeichern, sind Dateien auf der Diskette zu öffnen und nach dem Abspeichern zu schließen. Dies geschieht mit den Anweisungen OPEN und CLOSE.

Beispiel:

Vier Zahlenwerte sollen in einer Textdatei gespeichert werden.

```
10 PRINT CHR$(4);"OPEN ZAHLEN"
20 PRINT CHR$(4);"WRITE ZAHLEN"
30 PRINT "2000"
```

```
40 PRINT "3000"
50 PRINT "4000"
60 PRINT "5000"
70 PRINT CHR$(4);"CLOSE ZAHLEN"
```

In der Zeile 10 wurde die Textdatei ZAHLEN eröffnet. In Zeile 20 werden alle anschließenden Ausgabedaten in Form von PRINT-Anweisungen in die Textdatei ZAHLEN geschrieben. Die Zeilen 30 bis 40 kommen zur Ausführung, und Zeile 70 schließt die Datei ZAHLEN ab. Bei der Abarbeitung des Programms erscheint keine Ausgabe am Bildschirm. Wollen Sie die Prozedur verfolgen, muß der Befehl MON angewendet werden. Mit folgendem Programm können die Textdaten wieder gelesen werden.

```
10 PRINT CHR$(4);"OPEN ZAHLEN"
20 PRINT CHR$(4);"READ ZAHLEN"
30 FOR I = 1 TO 4
40 INPUT X$(I)
50 NEXT I
60 PRINT CHR$(4);"CLOSE ZAHLEN"
```

Den Textvariablen X$1 bis X$4 werden die vier Zahlenwerte zugewiesen. Ist MON eingeschaltet, wird folgende Ausgabe angezeigt:

```
OPEN ZAHLEN
READ ZAHLEN
?2000
?3000
?4000
?5000
CLOSE ZAHLEN
```

## APPEND

Diese Anweisung verknüpft zwei sequentielle Textfiles miteinander. Hinter Append muß der Schreibbefehl WRITE folgen. Wir wollen an unser letztes Beispiel zwei weitere Zahlen anhängen.

```
10 PRINT CHR$(4);"APPEND ZAHLEN"
20 PRINT CHR$(4);"WRITE ZAHLEN"
30 PRINT "6000"
40 PRINT "7000"
50 PRINT CHR$(4);"CLOSE ZAHLEN"
```

Wird unser Beispiel, das wir zum Lesen dieser Datei benutzt haben, an die zwei weiteren Zahlen angepaßt, dann können Sie sechs Zahlen lesen. Mit der Anweisung

**POSITION**

kann ein ganz bestimmtes Textsegment aus der Textdatei gelesen werden. Diese Anweisung muß folgende Syntax haben:

POSITION,Rz

Der Buchstabe z wird durch eine Zahl ersetzt.

### 8.5.2 Erstellen einer EXEC-Datei

Die Anweisung hat den großen Vorteil, daß damit eine ganze Programmsteuerung ohne manuellen Zugriff ablaufen kann. Damit können Steuerprogramme als Textdateien gespeichert werden.

Beispiel:

```
10 PRINT CHR$(4);"OPEN STEUERPROGRAMM"
20 PRINT CHR$(4);"WRITE STEUERPROGRAMM"
30 PRINT "CATALOG"
40 PRINT "RUN DEMO"
50 PRINT "LOAD TEST"
60 PRINT CHR$(4);"CLOSE STEUERPROGRAMM"
```

So kann eine gesamte Steuerdatei erstellt werden. Die Anweisungen müssen nur noch in PRINT-Anweisungen gefaßt sein. Ist das Programm abgelaufen, starten Sie mit

**EXEC STEUERPROGRAMM**

Das Textfile wird jetzt schrittweise abgearbeitet. Zuerst wird das Inhaltsverzeichnis der Diskette abgefragt und ausgegeben. Dann startet das Programm DEMO und wird anschließend durch das Programm TEST überschrieben.

### 8.5.3 Textdateien mit wahlfreiem Zugriff

Bei sequentiellen Dateien hatten wir das Problem, Daten, die wir in einer Textdatei abgespeichert hatten, schnell wiederzufinden. Mit Random-access-Dateien, wie wir Textdateien mit wahlfreiem Zugriff auch nennen, ist das leichter zu bewerkstelligen. Der Nachteil liegt im höheren Speicherverbrauch auf der Diskette, denn Dateien mit wahlfreiem Zugriff haben nach einer Definition alle die gleiche Größe. Besonders für die Erstellung einer Lagerkartei, einer Adreßverwaltung oder für ein Telefonregister ist diese Art von Dateien geeignet. Der einzige Unterschied zu sequentiellen Datei ist der, daß die Feldlängen angegeben werden. Sind mehrere Daten zusammengefaßt, spricht man von einem **Record.** Beim Abspeichern in eine Datei muß daher eine Recordnummer vergeben werden. Im folgenden Beispiel wollen wir ein Telefonregister erstellen. Dabei sollen der Name, die Telefonnummer und die Straße eingegeben werden.

Beispiel:

```
10 INPUT "NAME: ";A$
20 INPUT "TELEFON-NR.: ";T$
30 INPUT "STRASSE: ";S$
40 PRINT$(4);"OPEN TELEFON,L100"
50 PRINT$(4);"WRITE TELEFON,R1"
60 PRINTA$: PRINT T$: PRINT S$
70 PRINT CHR$(4);"CLOSE TELEFON"
```

Die INPUT-Anweisungen ordnen den Variablen A$,T$ und S$ die Texte zu, die anschließend in die Textdatei geschrieben werden. In Zeile 40 wird die Textdatei TELEFON geöffnet. L100 zeigt die Recordlänge an. In unserem Beispiel sind das 100 Zeichen, die Sie maximal eingeben können. Mit R1 wurde angezeigt, daß in Record 1 anschließend geschrieben wird. Textdateien sind im Inhaltsverzeichnis der Diskette mit T bezeichnet.

Beispiel:

```
10 PRINT CHR$(4);"OPEN TELEFON,L100"
20 PRINT CHR$(4);"READ TELEFON,R1"
30 INPUT A1$,T1$,S1$
40 PRINT CHR$(4);"CLOSE TELEFON"
```

Wird auch die Recordnummer variabel gestaltet, können Sie mehrere Telefonnummern speichern und auslesen.

### 8.5.4 Übertragen von Binärdaten

Schon bei der Grafikprogrammierung haben wir einen Speicherbereich auf Diskette geladen. Für fortgeschrittene Programmierer ist das der einzige Weg, Maschinenprogramme zu speichern oder in den Computer zu laden. Dazu benutzen wir die Anweisungen BLOAD und BSAVE.

**BSAVE**

Wird ein bestimmter Speicherbereich mit Binärdaten belegt und sollen diese auf Diskette gespeichert werden, ist der Befehl BSAVE anzuwenden. Dabei muß die Startadresse angegeben sein, ab der gespeichert werden soll. Anschließend muß die Länge des Speicherblockes gekennzeichnet werden. Die Angaben sind dezimal oder hexadezimal einzugeben. Nach dem Befehlswort BSAVE wird ein Dateiname vergeben. Nach einem Komma folgt der Buchstabe A für Anfang, danach die Startadresse, ab der gespeichert wird. Ist die Zahl hexadezimal, wird ein Dollarzeichen vorangestellt. Jetzt folgt hinter einem Komma der Buchstabe L für Länge. Danach die Länge des Speicherbereiches. Folgende Syntax ist zu beachten:

```
BSAVE Name,A.....,L......
```

Binärdateien sind auf der Diskette mit B gekennzeichnet. Soll der binäre Datenblock geladen werden, erfolgt der gleiche Befehlssatz, nur mit dem Befehlswort

**BSAVE**

Beispiel:

Speichern der Grafikseite 1

```
BSAVE BILD1,A$2000,L$2000
```

Laden der Grafikseite 2

BLOAD BILD2,A$4000,L$2000

Die Daten können auch in einen anderen Speicherbereich geladen werden. Dazu geben wir nur eine neue Anfangsspeicherstelle ein. Die Grafikseiten sind binäre Datenblöcke, die keine Programmfunktion ausüben können. Handelt es sich um ein Maschinenprogramm, können Sie das Programm mit

**BRUN**

laden, und DOS startet es automatisch.

Sollten Sie noch mehr über Dateiverwaltung und Diskettenbetrieb unter DOS wissen wollen, so empfehle ich Ihnen das DOS-Handbuch von Apple. In diesem Buch sind noch weitere Informationen enthalten.

## Was Sie über DOS 3.3 wissen sollten

Zum Schluß ein paar Fragen zum vergangenen Kapitel. Bitte beantworten Sie die Fragen ehrlich, und lesen Sie nochmals nach, wenn Sie etwas nicht verstanden haben.

1. Mit welchem Befehl werden BASIC-Programme gespeichert? Was ist dabei zu beachten?

2. Wie können Daten im Arbeitsspeicher mit den Informationen auf der Diskette verglichen werden?

3. Ist es möglich, Daten auf der Diskette zu schützen?

4. Welche Syntax hat der Befehl MON? Wie wird er ausgeschaltet?

5. Welchen Befehl verwenden wir zur Initialisierung einer Diskette? Was ist Initialisieren?

6. Womit werden Programme umbenannt oder gelöscht?

# 9

# Das Betriebssystem ProDOS

# 9 Das Betriebssystem ProDOS

Im letzten Kapitel haben wir uns ausführlich mit DOS 3.3 beschäftigt. Wir wissen, was mit einem Betriebssystem gemeint ist und kennen seine Aufgaben und Möglichkeiten. Aber DOS hat einige Nachteile, die eine Verbesserung oder ein neues Betriebssystem erforderten. Das umständliche Einbinden in Maschinenroutinen, aber auch der ansprechbare Speicherplatz hat bei DOS seine Grenzen. Aus diesen und anderen Gründen entwickelte man bei Apple ein neues Betriebssystem für die Apple-II-Serie. Es trägt den Namen ProDOS. Mit ihm wird es möglich, auch größere Datenmengen zu verarbeiten. Eine spätere Adaptierung an ein Festplattenlaufwerk ist damit vorprogrammiert. ProDOS unterstützt außerdem die vollen 128 KByte Arbeitsspeicher des Apple IIc.

## 9.1 Was ist ProDOS?

ProDOS ist die Abkürzung von Professional Disk Operating System. Zu deutsch: Professionelles Diskettenverwaltungsprogramm. Es handelt sich dabei um ein Programm, das den Datenaustausch zwischen Computer und Datenträger verwaltet. Das kann ein Disketten- , aber auch ein Festplattenlaufwerk sein. Ein Festplattenlaufwerk besteht aus übereinanderliegenden, magnetisierbaren Platten, die von Schreib-/Leseköpfen abgetastet werden. Die Schreibdichte ist durch die exakte Positionierung der Schreib-/Leseköpfe auf den Platten höher als bei Diskettenlaufwerken. Um ProDOS voll ausnutzen zu können, müßte ein solcher Massenspeicher vorhanden sein. Denn ProDOS ist ein Betriebssystem, das große Datenmengen strukturiert verwalten kann und den Datenaustausch zwischen Computer und Magnetspeicher koordiniert. Die Lade- und Speicherroutinen sind schneller und umfangreicher gestaltet. Dadurch kann ein Programm in kürzerer Zeit geladen oder gespeichert werden, als dies mit DOS der Fall ist. Durch den strukturierten Aufbau wird ein rasches Finden von Datenbeständen ermöglicht.

## 9.2 Laden von ProDOS

Legen Sie die Diskette "System-Dienstprogramme" in das Laufwerk Ihres Apple IIc. Nach dem Einschalten dauert es zirka 30 Sekunden, bis die Programmteile geladen sind, dann zeigt der Bildschirm ein Menü mit 9 Punkten an. So finden Sie unter jedem Menüpunkt die gewünschten Programme. Um das Programm mit den Hilfsroutinen verlassen zu können, wählen Sie den Menüpunkt 9. Mit der Pfeil-nach-oben- oder der Pfeil-nach-unten-Taste erfolgt diese Auswahl. Dabei zeigen die Zeichen < > das ausgewählte Unterprogramm an. Haben Sie den Menüpunkt 9 gewählt und mit RETURN beendet, verlassen Sie das Programm bzw. das Auswahlmenü. Sie werden aber vorsichtshalber nochmals vom Programm gefragt, ob Sie auch wirklich das Programm verlassen wollen. Bild 9-1 zeigt uns die Anzeige am Bildschirm. Sollten Sie nochmals RETURN drükken, wird der Bildschirm gelöscht. Die 80-Zeichendarstellung wird deaktiviert, und Sie befinden sich im Applesoft-BASIC-Modus. Der Cursor blinkt hinter einem großen Ü und zeigt die Bereitschaft an, Applesoft-BASIC-Programme entgegenzunehmen.

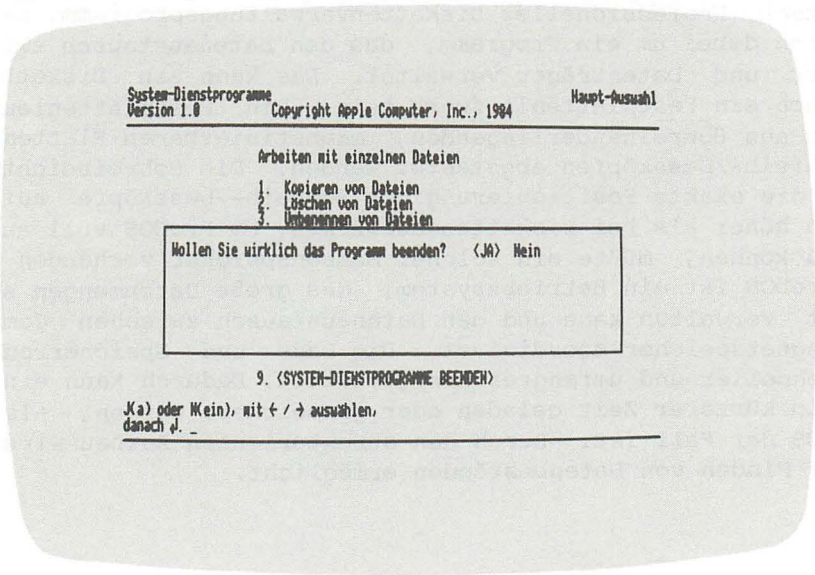



Abb. 9-1: Verlassen der System-Dienstprogramme

Durch das Starten der Diskette "System-Dienstprogramme" wurde das Betriebssystem ProDOS geladen. Alle Anweisungen, die einen Diskettenzugriff ermöglichen, laufen nun unter der Verwaltung von ProDOS. Darum sind auch die Befehle, die wir bei DOS gelernt haben, nur noch zum Teil gültig. ProDOS verwendet nur einen Teil der Anweisungen, die wir von DOS her kennen. Einige Befehle haben eine andere Bedeutung bekommen.

## 9.3 System-Dienstprogramme

Die Diskette "System-Dienstprogramme" beinhaltet Programme, die uns das Arbeiten mit Disketten wesentlich erleichtert. Im folgenden Abschnitt wollen wir deshalb die Programme genauer anschauen, denn sie können uns eine Menge Arbeit ersparen. Die Unterprogramme werden der Reihe nach von oben nach unten besprochen. Die Überschriften der folgenden Abschnitte zeigen den jeweiligen Programmpunkt an.

### 9.3.1 Kopieren von Dateien

Zum Kopieren von Dateien haben wir den Punkt 1 ausgewählt. Im Untermenü, im Unterprogramm "Kopieren von Dateien", stehen uns drei Möglichkeiten zur Verfügung, verschiedene Teile des Hilfsprogramms zu aktivieren. Wir werden nach dem Laufwerk gefragt, in welchem sich die Diskette befindet, die kopiert werden soll. In den meisten Fällen ist es das eingebaute Laufwerk. Wir drücken also RETURN, denn die Zeiger stehen bei "EINGEBAUTES LAUFWERK". Jetzt wird nach dem Ziellaufwerk gefragt. Als Ziellaufwerk wird das Laufwerk bezeichnet, in das die Datei gespeichert werden soll. Das interne Laufwerk ist auch unser Ziellaufwerk. So, schon wieder ein neues Menü. Sollen alle oder nur einige Dateien kopiert werden? Wir wählen den Menüpunkt EINIGE. Das Diskettenlaufwerk liest das Directory (Inhaltsverzeichnis) und markiert das erste Programmfile (File = Name). Mit den angezeigten Tasten können wir die Dateien, die wir kopieren wollen, markieren. Ist die Auswahl getroffen und sind die Dateien markiert, kann mit RETURN gestartet werden. Das Diskettenlaufwerk liest so viele Programme in den Speicher ein, bis dieser voll ist, dannach werden wir zum

Diskettenwechsel aufgefordert. Die Diskette, auf die das Programm kopiert werden soll, muß im gleichen Format angelegt sein wie die Originaldiskette. Jetzt schreibt der Schreib-/Lesekopf des Diskettenlaufwerks die Daten, die eingelesen wurden, auf die Zieldiskette. Das Programm gibt am Bildschirm die Dateinamen aus, die kopiert wurden. Ist eine Datei auf die neue Diskette geschrieben, wird das Wort "beendet" ausgegeben und die nächste Datei gelesen. Der Kopiervorgang ist beendet, wenn "Kopieren beendet" am unteren Bildschirmrand erscheint. Mit RETURN können Sie erneut Dateien kopieren. Mit der ESCape-Taste kommen Sie zum Hauptmenü zurück.

### 9.3.2 Löschen von Dateien

Wie schon beim Hauptmenüpunkt 1, so werden wir auch beim Löschen von Dateien nach den einzelnen Laufwerken gefragt. Geben Sie das Laufwerk an, in welchem sich die Datei befindet, die gelöscht werden soll. Nach dem Drücken der RETURN-Taste liest das Laufwerk das Inhaltsverzeichnis der Diskette und gibt es am Bildschirm aus. Wählen Sie die Datei, die Sie löschen wollen. Das Programm überprüft, ob die Datei zu löschen ist. Das trifft dann zu, wenn die Datei softwaregeschützt ist oder die Diskette nicht beschrieben werden kann. Dabei wird das Wort "storniert" hinter dem Dateinamen am Bildschirm ausgegeben. Wie schon beim Kopieren von Dateien, ist der Löschvorgang der Dateien dann abgeschlossen, wenn am unteren Bildschirmrand "Löschen beendet" erscheint.

### 9.3.3 Umbenennen von Dateien

Auch bei diesem Menüpunkt werden Sie nach dem Laufwerk gefragt, in welchem sich die Datei zum Umbenennen befindet. Hat das Laufwerk das Inhaltsverzeichnis mit den Dateien geladen, wählen Sie die Datei, die umbenannt werden soll. Die markierten Dateien werden nochmals gesondert angezeigt, wobei nach dem neuen Namen gefragt wird. Ist der Dateiname mit dem angegebenen identisch, erfolgt kein Zugriff auf das Diskettenlaufwerk. Ist ein neuer Name angegeben worden, wird dieser nur dann geändert, wenn auf die Diskette zugegriffen werden kann. Das heißt, es darf sich kein Schreibschutz an der Diskette befinden. Ist die Datei software-

geschützt, wird vom Programm gefragt, ob die Datei umbenannt werden soll. Wird "NEIN" angegeben, wird das Umbenennen storniert.

### 9.3.4 Schreibschutz setzen/aufheben

Mit dem Befehl LOCK können wir Dateien gegen Überschreiben schützen. Die gleiche Funktion wird von Menüpunkt 4 erfüllt. Der Schreibschutz kann nur dann gesetzt werden, wenn die Diskette nicht mechanisch schreibgeschützt ist, das heißt die Schreibkerbe frei und nicht mit einem Klebeetikett verschlossen.

### 9.3.5 Kopieren einer Diskette

Der Menüpunkt 5 gehört zur Kategorie der Programmpunkte, die die ganze Diskette ansprechen. Mit diesem Unterprogramm kann eine ganze Diskette kopiert werden, um z.B. eine Sicherungskopie Ihrer Programmdiskette zu erstellen. Kommerzielle Disketten dagegen sind nicht nur mit LOCK geschützt. Sie haben raffinierte Schutzvorrichtungen, die nur schwer zu umgehen sind. Außerdem ist das Kopieren dieser Disketten grundsätzlich nicht erlaubt, und Sie würden sich strafbar machen. Bei den Disketten, die kopiert werden sollen, muß es sich um ungeschützte Disketten handeln.

Beim Kopieren zeigt sich der Vorteil, wenn Sie ein externes Diskettenlaufwerk besitzen. Sie legen die Originaldisketten in das interne und die Kopiediskette in das externe Laufwerk. Bei einem Laufwerk ist ein umfangreicher Diskettenwechsel erforderlich. Sollte die Diskette, auf die geschrieben werden soll, noch nicht formatiert sein, so übernimmt das Programm automatische den Formatiervorgang und zeigt die gewählten Laufwerke sowie das Format der Originaldiskette am Bildschirm an. Soll eine Backup (Kopie) der Diskette gemacht werden, ohne den Diskettennamen zu ändern, wird der angegebene Originalname mit RETURN abgeschlossen. Das Programm übernimmt den Originalnamen auf die Kopie. Wird versucht, auf eine bereits beschriebene Diskette zu kopieren, weist das Programm Sie darauf hin und bietet die Möglichkeit, das Kopierprogramm zu verlassen.

### 9.3.6 Formatieren einer Diskette

Mit diesem Unterprogramm kann die Diskette formatiert werden. Sie können dabei auswählen zwischen ProDOS-, DOS- und Pascalformat. Wissen Sie nicht, welches Format Sie verwenden sollen, so können Sie eine entsprechend formatierte Diskette einlegen. Der Computer analysiert das Format und legt es fest. Beim Format ProDOS und Pascal müssen Sie einen Diskettennamen vergeben. Dieser ist zur späteren Selektierung notwendig.

Bei diesem Menüpunkt ist zu beachten, daß nur das Format angelegt wird. Die formatierten Disketten lassen sich nur dann starten, wenn sich das Betriebssystem DOS 3.3 auf der Diskette befindet. Die anderen Formate müssen zuerst mit Start- oder Systemprogrammen versehen werden.

### 9.3.7 Katalog einer Diskette

Genau wie der Befehl CATALOG in DOS kann auch in diesem Unterprogramm das Inhaltsverzeichnis einer Diskette angezeigt werden. Darüber hinaus sehen Sie den Diskettennamen und das Diskettenformat. Außerdem werden die Dateiformen genau spezifiziert. Die Anzahl der Dateien und die belegten Blöcke werden genauso angezeigt wie die Blöcke, die noch frei sind. Mit diesem Unterprogramm ist es auch möglich, das Inhaltsverzeichnis einer Diskette auf einem Drucker auszugeben, wobei der entsprechende Anschluß angegeben werden muß. Ein Ausdruck kann bei der Suche einer Datei auf mehreren Disketten sehr nützlich sein. Das ausgedruckte Inhaltsverzeichnis sollte jeder Diskette beigelegt sein. Abbildung 9-2 zeigt den Ausdruck des Inhaltsverzeichnisses der System-Dienstprogramm-Diskette.

```
ücatalog

/UTILITIES

 NAME            TYPE  BLOCKS  MODIFIED          CREATED           ENDFILE SUBTYPE

 USER.PROFILE    TXT        1  <NO DATE>         <NO DATE>               0 R=     0
*STARTUP         BAS        4   2-APR-84  0:00   <NO DATE>            1176
*SU              BAS       35   2-APR-84  0:00   <NO DATE>           16999
*SU1.OBJ         BIN       28   2-APR-84  0:00   <NO DATE>           13427 A=$3200
*SU2.OBJ         BIN       10  15-MAR-84  0:00   <NO DATE>            4608 A=$2000
*SU3.OBJ         BIN       61   2-APR-84  0:00   <NO DATE>           30682 A=$0F00
*SU4.OBJ         VAR       19   2-APR-84  0:00   <NO DATE>            8767
*PRODOS          SYS       31  15-MAR-84  9:03   <NO DATE>           15360
*BASIC.SYSTEM    SYS       21  15-MAR-84  0:00   <NO DATE>           10240

BLOCKS FREE:   63     BLOCKS USED:  217     TOTAL BLOCKS:  280
```

Abb. 9-2: Inhaltsverzeichnis der System-Dienstprogramm-Diskette

### 9.3.8 Sonstige Funktionen

Dieser Programmpunkt verzweigt in ein weiteres wichtiges Menü. Dabei können wir zwischen ProDOS-Unterprogrammen und weiteren Funktionen wählen.

### 9.3.9 PREFIX setzen

Damit kommen wir zu einer Besonderheit von ProDOS. Als Prefix bezeichnet man einen Dateinamen, der eine ganz bestimmte Diskette anspricht. Die System-Dienstprogramme-Diskette hat das ProDOS-Format und besitzt den Volumennamen UTILITIES. Dieser Volumenname wird als PREFIX beim Starten der Diskette gesetzt. Soll eine andere Diskette angesprochen werden, so muß der PREFIX neu gesetzt werden. Dabei ist UTILITIES der PREFIX der Diskette. Ein weiterer

Name würde dann der Filename oder ein Unterinhaltsverzeichnis sein. Wir sprechen dann von einem Subdirectory. Der PREFIX ist der Zeiger eines Pfadnamens, der anzeigt, welche Diskette verwendet wird. Ihm folgt, durch einen Schrägstrich getrennt, der Subdirectoryname, sofern ein Unterinhaltsverzeichnis vorhanden ist. Wieder durch einen Schrägstrich getrennt folgt dann der Filename (Programm- oder Dateiname). Diese tiefe Verschachtelung ist nur dann sinnvoll, wenn ein Massenspeicher zur Verfügung steht. Das wäre zum Beispiel ein Festplattenlaufwerk mit einer Speicherkapazität von 5000 KByte, also 5 MegaByte.

Beispiel:

/Lebensmittel/Gemüse/Bohnen

Dieses Beispiel soll den Aufbau eines **Pfadnamens** erklären. Die "Lebensmittel" sind der PREFIX. "Bohnen" ist der Dateiname, der unter dem Subdirectory "Gemüse" abgelegt ist. An diesem Beispiel kann man erkennen, daß es in ProDOS eine Vielzahl von Verzweigungen gibt. Dadurch ist es möglich, Dateien bestimmten Kategorien zuzuordnen, die dann in einem Unterinhaltsverzeichnis (Subdirectory) zusammengefaßt sind.

### 9.3.10 Subdirectory erstellen

In einem Subdirectory können Dateinamen abgelegt werden. Das ermöglicht es uns, ein Programm mit seinen Teilprogrammen darin zusammenzufassen. Im Menü "Sonstige Funktionen" können wir ein Subdirectory erstellen. Dazu müssen Sie nur einen Namen eingeben. Wird der Name mit RETURN abgeschlossen, schreibt das Laufwerk den neuen Subdirectorynamen auf die Diskette. Beim Auflisten des Inhaltsverzeichnisses sehen wir, daß das Subdirectory die Typenbezeichnung "Dir" erhalten hat.

### 9.3.11 Konvertieren des Diskettenformats

Mit diesem Programmpunkt kann eine DOS-3.3-Diskette in eine ProDOS-Diskette umgewandelt werden. Auch können Sie eine DOS-3.2-Diskette, die mit 13 Sektoren arbeitet, auf das 16-Sektoren-Format der DOS-3.3-Disketten konvertieren. Ebenfalls ist eine Konvertierung von ProDOS auf DOS 3.3 möglich. Bei der Umformung werden die Befehle in den Programmen nicht geändert. Es wird ausschließlich das Format geändert. Die Aufzeichnungen selbst bleiben dabei unberührt.

### 9.3.12 Überprüfen einer Diskette

Mit dieser Funktion kann die Diskette auf ihre Lesbarkeit überprüft werden. Denn Disketten können schon bei der Herstellung Fehler aufweisen. Auch beschriebene Disketten müssen von Zeit zu Zeit überprüft werden. Sollte eine Diskette Fehler beinhalten, sind die Dateien auf eine funktionsfähige Diskette zu übertragen. Eine fehlerhafte Diskette sollte neu formatiert werden. Liegt danach immer noch ein Fehler vor, dann ist sie unbrauchbar.

### 9.3.13 Serielle Anschlüsse konfigurieren

Soll ein Peripheriegerät angeschlossen werden, das nicht standardmäßig mit dem Apple IIc verbunden werden kann, muß dieser Programmpunkt ausgewählt werden. Es können dann auch Drucker, Plotter oder Modem angeschlossen werden, die keine Standardbelegung haben. Die Modifikationen werden anschließend auf der Diskette festgehalten, d.h., die seriellen Anschlüsse müssen nur einmal konfiguriert werden.

Damit sind alle Möglichkeiten der Diskette "System-Dienstprogramme" erläutert worden. Im weiteren Abschnitt werden jetzt die spezifischen Eigenschaften von ProDOS erklärt.

## 9.4 Anzeigen des Inhaltsverzeichnisses in ProDOS

Auch in BASIC ist es möglich, ProDOS zu verwenden. Sie sollten bald auf dieses Betriebssystem umsteigen, wenn Sie schon mit DOS gearbeitet haben. Um den Inhalt einer Diskette anzuzeigen, haben wir in DOS den Befehl **CATALOG** angewendet. Er ist auch in ProDOS gültig. In der 80-Zeichendarstellung zeigt er uns das Inhaltsverzeichnis genau an. Wir sehen den Filenamen, Subdirectorys, den Filetyp, die belegten Blocks und einige andere Daten. Ist z.B. ein Uhrenmodul (Real-time-clock) angeschlossen, so werden die Zeit und das Datum der Dateierstellung oder das Datum der Modifizierung in das Inhaltsverzeichnis eingetragen. In den letzten beiden Spalten steht die Größe der Dateien und bei Binärdaten deren Anfangsadresse beim Abspeichern.

```
ücatalog

/UTILITIES

 NAME          TYPE  BLOCKS  MODIFIED          CREATED          ENDFILE SUBTYPE

 USER.PROFILE  TXT      1    <NO DATE>         <NO DATE>              0 R=     0
*STARTUP       BAS      4    2-APR-84  0:00    <NO DATE>           1176
*SU            BAS     35    2-APR-84  0:00    <NO DATE>          16999
*SU1.OBJ       BIN     28    2-APR-84  0:00    <NO DATE>          13427 A=$3200
*SU2.OBJ       BIN     10   15-MAR-84  0:00    <NO DATE>           4608 A=$2000
*SU3.OBJ       BIN     61    2-APR-84  0:00    <NO DATE>          30682 A=$0F00
*SU4.OBJ       VAR     19    2-APR-84  0:00    <NO DATE>           8767
*PRODOS        SYS     31   15-MAR-84  9:03    <NO DATE>          15360
*BASIC.SYSTEM  SYS     21   15-MAR-84  0:00    <NO DATE>          10240

BLOCKS FREE:  63    BLOCKS USED:  217    TOTAL BLOCKS:  280
```

Abb. 9-3: Directory mit CATALOG

Im 40-Zeichen-Modus wird der Befehl anders formuliert. Bei der Eingabe von **CAT** wird auch das Inhaltsverzeichnis der Diskette ausgegeben. Dabei werden die letzten vier Spalten, die mit CATALOG ausgegeben wurden, unterdrückt. Abbildung 9-4 zeigt uns das veränderte Inhaltsverzeichnis. Die Anweisungen CAT und CATALOG werden in der Betriebsart Direktanweisung verwendet. Sie sind aber auch in Programmen einsetzbar. Hinter die Anweisungen kann die Laufwerksangabe, durch ein Komma getrennt, angefügt werden.

Dabei selektiert D1 das interne Laufwerk und D2 das externe Laufwerk. Wird ein Laufwerk mit einer höheren Gerätezahl gewählt, erscheint die Fehlermeldung

RANGE ERROR

Da nur zwei Laufwerke maximal zur Verfügung stehen, darf nur D1 und D2 verwendet werden. Wurde z.B. CAT,D2 eingegeben, wird das Inhaltsverzeichnis der Diskette im externen Laufwerk ausgegeben. Solange nicht Laufwerk D1 angesprochen wird oder ein System-Reset erfolgt ist, werden alle Laufwerksfunktionen zum externen Laufwerk geleitet. Um das interne Laufwerk wieder anzusprechen, muß dieses mit D1 selektiert werden.

```
ücat

/UTILITIES

 NAME            TYPE  BLOCKS  MODIFIED

 USER.PROFILE     TXT       1  <NO DATE>
*STARTUP          BAS       4   2-APR-84
*SU               BAS      35   2-APR-84
*SU1.OBJ          BIN      28   2-APR-84
*SU2.OBJ          BIN      10  15-MAR-84
*SU3.OBJ          BIN      61   2-APR-84
*SU4.OBJ          VAR      19   2-APR-84
*PRODOS           SYS      31  15-MAR-84
*BASIC.SYSTEM     SYS      21  15-MAR-84

BLOCKS FREE:   63     BLOCKS USED:  217
```

Abb. 9-4: Directory mit CAT

In der Spalte TYPE wird der Filetype der Datei angegeben. Folgende Aufstellung zeigt die Spezifizierung und die Unterschiede zu DOS.

Tabelle 9-1: Filetypen in DOS und ProDOS

| DOS | ProDOS | Filetype |
|---|---|---|
| - | $00 | Typeless File |
| - | $12 - $BF | SOS Reserved |
| - | $F1 - $F8 | ProDOS User Defined File 0-9 |
| - | $F9 | ProDOS Reserved |
| - | BA3 | Business Basic Program File |
| - | BAD | Bad Block File |
| A | BAS | Applesoft Programfile |
| B | BIN | Binär File |
| - | CO - EF | ProDOS Reserved |
| - | CMD | ProDOS Added Command File |
| - | DA3 | Business Basic Data File |
| - | DIR | Directory |
| - | FNT | Font File |
| - | FOT | Graphics Screen File |
| I | INT | Integer-BASIC Program |
| - | IVR | Integer-BASIC Variable |
| - | PCD | Pascal Code |
| - | PDA | Pascal Data |
| - | PTX | Pascal Text |
| R | REL | Relocatable Code |
| - | RPD | RPS Data File |
| - | RPI | RPS Index File |
| - | SOS | Apple III System File |
| - | SYS | ProDOS System File |
| T | TXT | ASCII-Text-File |
| - | VAR | Applesoft Variable |
| - | WPF | Word Processor File |

## 9.5 Startup-Programm

Beim Starten einer Diskette wurde in DOS das Programm HELLO geladen und anschließend gestartet. In ProDOS muß die Datei oder der Programmname STARTUP auf der Diskette vorhanden sein. Dabei können verschiedene Dateitypen diesen Namen tragen. Wird die Diskette gebootet (gestartet), dann werden die Daten oder das Programm in der Datei STARTUP abgearbeitet. Auf einer Startdiskette im ProDOS-Format müssen die Datei oder das Programm STARTUP sowie das ProDOS- und BASIC-Systemfile vorhanden sein.

**Für Profis!**

Wird die Diskette mit den drei Programm- und Datenfiles gestartet, transferiert das Controllerprogramm des Computers den Inhalt auf der Diskette ab Track 0 in den Arbeitsspeicher ab der Speicheradresse $0800 oder startet dort. Das Controllerprogramm befindet sich in der Steuerelektronik des Diskettenlaufwerks. Das Startprogramm, das in einem ROM gespeichert ist, sucht nun auf der Diskette in Laufwerk 1 das Systemfile PRODOS. Ist das File vom Startprogramm nicht auffindbar, erscheint die Fehlermeldung

**UNABLE TO LOAD PRODOS**

Ist PRODOS.SYS vorhanden, wird es parallel zum Monitor ab der Speicheradresse $F000 abgelegt. Jetzt übernimmt das ProDOS-Programm den weiteren Ablauf des Systemstarts. Es sucht auf der Diskette in Laufwerk 1 ein File mit der Typenbezeichnung (Suffix) "System", abgekürzt SYS. In den meisten Fällen handelt es sich dabei um das BASIC.SYSTEM. Die Systemdaten von BASIC.SYSTEM werden in den Speicherbereich zwischen $9A00 und $BFFF geladen. Dieser Bereich wurde auch von DOS verwendet. Die ProDOS-Anweisungen werden jetzt in diesem Speicherbereich bearbeitet. BASIC.SYSTEM dient als Koordinator zwischen dem BASIC-Interpreter und ProDOS arbeitet. Jetzt sucht ProDOS nach einem Programm oder einer Datei mit dem Namen STARTUP. Wird es gefunden, startet der Computer das Programm. Ist STARTUP nicht auffindbar, wird automatisch in den BASIC-Interpreter gesprungen. Der Computer gibt folgende Anzeige auf dem Bildschirm aus:

**PRODOS BASIC 1.0**
**COPYRIGHT APPLE, 1983**

Wird auf einer Diskette kein BASIC.SYSTEM und kein PRODOS gespeichert, kann die Diskette als reine Datendiskette benutzt werden, sofern sie das ProDOS-Format besitzt.

## 9.6 Laden und Abspeichern von Daten

Die Hauptaufgabe eines Betriebssystems ist die Übertragung von Daten von und zu Datenträgern. Für die Speicherung von BASIC-Programmen hatten wir in DOS den Befehl **SAVE** verwendet. Auch in ProDOS kann dieser Befehl in gleicher Form wieder gebraucht werden. Das gleiche gilt auch für das Laden von BASIC-Programmen. Die Anweisung **LOAD** wird in ProDOS wie in DOS benützt. Zusätzlich gibt es aber in ProDOS noch die Möglichkeit, mit dem Trennstrich (-) Daten von der Diskette zu laden. Dabei spielt es keine Rolle, um welchen Datentyp es sich handelt. Im Normalfall würden wir schreiben:

LOAD BEISPIEL

Wir können aber auch durch Verwendung des Trennstriches Daten in den Computer übertragen. Wir müssen dazu statt der Anweisung LOAD den Trennstrich setzen. Dazu wandeln wir das Beispiel folgendermaßen ab:

- BEISPIELE

So muß nur der Programm- oder Dateiname angegeben werden. ProDOS lädt das Programm oder die Datei an die richtige Adresse im Arbeitsspeicher. Sollen Binärdaten geladen oder gespeichert werden, so können die Anweisungen BLOAD und BSAVE verwendet werden (vgl. DOS) Durch die Anzeige des Inhaltsverzeichnisses einer Diskette mit der Anweisung CATALOG können wir feststellen, mit welcher Anfangsadresse die binären Daten geladen wurden. Im Anzeigefeld SUBTYPE können wir die Anfangsadresse in hexadezimaler Darstellung ablesen.

Beispiel:

Eingabe:

```
BSAVE Seite1,A$2000,L$2000
BSAVE Seite2,A$4000,L$2000
```

Ausgabe mit der Anweisung CATALOG:

```
ücatalog

/UTILITIES

 NAME            TYPE  BLOCKS  MODIFIED          CREATED           ENDFILE SUBTYPE

 SEITE1           BIN      17  24-NOV-84 17:51   24-NOV-84 17:51      8192 A=$2000
*BASIC.SYSTEM     SYS      21  <NO DATE>         <NO DATE>           10240
 SEITE2           BIN      17  24-NOV-84 17:51   24-NOV-84 17:51      8192 A=$4000
*PRODOS           SYS      31  15-MAR-84  9:03   <NO DATE>           15360

BLOCKS FREE:  187     BLOCKS USED:   93     TOTAL BLOCKS:  280
```

Abb. 9-5: CATALOG mit Binärdaten

In diesem Beispiel haben wir die Daten, die in den Speicherbereichen der Grafikseiten 1 und 2 liegen, auf Diskette abgespeichert. Am CATALOG-Eintrag der Diskette können wir erkennen, daß die Grafikseite 1 ab der Speicherstelle $2000 gespeichert wurde. Die Grafikseite 2 ist ab der Speicherstelle $4000 gespeichert worden.

Ein großer Vorteil von ProDOS ist die Möglichkeit, alle Anweisungen in Groß- oder Kleinschrift eingeben zu können. Dabei spielt es keine Rolle, daß die Filenamen im Catalog in Großbuchstaben erscheinen. Wird ein neuer Filename vergeben, so wird dieser auch dann in Großschrift im Catalog abgelegt, wenn er in Kleinschrift eingegeben wurde.

## 9.7 FP und INT in ProDOS

Mit den Anweisungen FP und INT konnten wir von Applesoft-BASIC in Integer-BASIC umschalten und umgekehrt. Diese Anweisungen stehen uns unter dem Betriebssystem ProDOS leider nicht mehr zu Verfügung. Dadurch kann auch Integer-BASIC nur unter DOS benutzt werden. Wenn Sie die Anweisungen in ProDOS benutzen, erfolgt die Fehlermeldung

?SYNTAX ERROR.

## 9.8 Formatieren der Diskette

Im Betriebssystem DOS 3.3 mußte zum Formatieren einer Diskette die Anweisung INIT verwendet werden. ProDOS kennt diesen Formatierungsbefehl nicht. Die Formatierung muß immer über die Systemdiskette erfolgen.

## 9.9 Eingabeformat in ProDOS

In DOS war es möglich, Programm- oder Dateinamen und Anweisungen mit Leerzeichen einzugeben. ProDOS würde Dateinamen in dieser Eingabeform nicht finden, denn es verlangt eine Eingabe ohne Leerzeichen. CONTROL-Zeichen sind ebenso nicht erlaubt wie die Verwendung anderer Satzzeichen, außer dem Punkt. Die Länge der File- und Volumennamen ist auf 15 Zeichen begrenzt. Ein Pfadname darf nicht mehr als 64 Zeichen enthalten, und maximal können bis zu 51 Filenamen in einem Directory stehen.

## 9.10 Erstellen eines Subdirectory

Sollen Daten oder Unterprogramme einem Hauptprogramm zugeordnet werden, so kann es von Vorteil sein, diese zu ordnen. In ProDOS können wir einem Hauptprogramm viele Unterprogramme zuordnen, ohne daß sie im Directory erscheinen. Wir müssen dazu ein Sub-

directory erstellen. In diesem Unterinhaltsverzeichnis legen wir die Daten ab, die zu einem bestimmten Hauptbereich gehören. Ein Subdirectory wird mit

CREATE

erstellt. Als Beispiel wollen wir auf einer Diskette mit dem Volumennamen TEST ein Subdirectory erstellen. Dazu geben wir nach der Anweisung CREATE den Volumennamen und getrennt durch einen Schrägstrich die Subdirectorybezeichnung ein. Danach folgt der Programm- oder Dateiname, der wieder durch einen Schrägstrich getrennt dem Subdirectory zugeordnet ist.

Beispiel:

create/test/test1

Test ist der Volumenname der Diskette. In diesem Beispiel verwenden wir eine formatierte Diskette mit dem Volumennamen test, die keine weiteren Files enthält. Dadurch können wir die Struktur besser übersehen. Auf der Diskette test wird das Subdirectory test1 angelegt. Der Schrägstrich und der Volumenname zwischen create und dem ersten Subdirectory können entfallen, wenn der Prefix /test/ gesetzt wurde. Dazu geben wir die Befehlsfolge ein:

Prefix/test/

Wir können die Createanweisung wie folgt abändern:

create test1

Es kann immer nur ein neues Subdirectory erzeugt werden. Wollen Sie auf dieses zugreifen, müssen Sie den Pfadnamen eingeben. In unserem Beispiel lautet der Pfadname

catalog/test/test1

Soll im Subdirectory ein File abgelegt werden, so ist der Prefix neu zu setzen oder der Pfadname anzugeben. In unserem Beispiel wollen wir ein File mit dem Namen "filename" im Subdirectory test1 ablegen.

save/test/test1/filename

Wird diese Befehlsfolge mit RETURN abgeschlossen, sichert ProDOS den BASIC-Speicherinhalt im Subdirectory unter dem Filenamen "filename". Wir wollen uns die verschiedenen Directorys einmal genauer anschauen.

```
ücatalog

/TEST

 NAME          TYPE  BLOCKS  MODIFIED        CREATED         ENDFILE SUBTYPE

 TEST1          DIR       1  <NO DATE>       <NO DATE>           512

BLOCKS FREE:  271     BLOCKS USED:    9    TOTAL BLOCKS:  280
```

Abb. 9-6: CATALOG /TEST

Wir sehen im CATALOG-Ausdruck der Diskette, daß ein Filename mit der Bezeichnung test1 angelegt wurde. Im Feld TYPE erkennen wir, daß die Typenbezeichnung DIR eingetragen wurde. Obwohl das Programm filename auf der Diskette gespeichert ist, erscheint es nicht im Inhaltsverzeichnis. Erst das Inhaltsverzeichnis des Subdirectorys test1 zeigt den Programmnamen.

```
ücatalog/test/test1

TEST1

 NAME          TYPE  BLOCKS  MODIFIED        CREATED         ENDFILE SUBTYPE

 FILNAME        BAS       1  <NO DATE>       <NO DATE>            15

BLOCKS FREE:  271     BLOCKS USED:    9    TOTAL BLOCKS:  280
```

Abb. 9-7: CATALOG /TEST1

Mit der Anweisung catalog/test/test1 wurde das Directory des Subdirectorys Test1 ausgegeben. Die belegten und freien Blöcke bleiben dabei gleich und ändern sich nur mit dem Gesamtbereich. Wir könnten nun weitere Subdirectorys anlegen. Dies kann im Volumendirectory oder in einem Subdirectory erfolgen.

## 9.11 Aktivieren der RAM-Disk

Beim Einladen von ProDOS hat das Startprogramm einen Speicherbereich von 128 KByte festgestellt. Das ist die Voraussetzung für ProDOS, um einen Speicherbereich wie das Diskettenlaufwerk benutzen zu können. Dabei werden die Daten nicht auf einer Diskette abgespeichert, sondern im Arbeitsspeicher des Apple IIc abgelegt. Dieser Bereich wird vom Betriebssystem gegen Überschreiben geschützt. Die Pseudo-Disk ist in 120 Blöcke mit je 512 Byte aufgeteilt. Um dieses "Pseudo-Diskettenlaufwerk" aktivieren zu können, müssen Sie folgenden Prefix setzen:

prefix/ram

Jetzt werden alle ProDOS-Anweisungen zum RAM-Drive geleitet. Um das zu beweisen, geben Sie bitte den Befehl CAT ein. Sie sehen folgendes Directory:

```
ücat

/RAM

 NAME            TYPE  BLOCKS  MODIFIED


BLOCKS FREE:  120     BLOCKS USED:    8
```

Abb. 9-8: Directory des leeren RAM-Drives

In Programmen kann dieser Speicherbereich auch zur Zwischenablage von größeren Datenmengen benutzt werden. Das Pseudo-Laufwerk ist als Drive2/Slot3 ansprechbar. Durch das völlige Fehlen einer Mechanik kann eine äußerst schnelle Übertragung erfolgen.

## 9.12 PR# und IN# in ProDOS

Eine elegante Lösung bietet ProDOS bei der Ansprache von I/O-Vektoren, um Peripheriegeräte anzusprechen. In ProDOS können Sie den Befehlen PR# und IN# eine Startadresse anhängen, die es ermöglicht, Unterprogramme in Maschinensprache aufzurufen. Dazu muß nur die Startadresse in hexadezimaler Form angefügt werden.

Beispiel:

```
PR#A$300
```

Findet ProDOS diese Anweisung in einem BASIC-Programm, springt der Computer an die Speicherstelle $0300 und arbeitet das Programm ab, das ab dieser Speicherstelle beginnt.

## 9.13 Filepuffer bei der Datenübertragung

Für die Datenübertragung von und zur Diskette bei der Bearbeitung von sequentiellen Dateien benutzt ProDOS zur Zwischenspeicherung der ankommenden Daten einen File-Buffer. Dieser besitzt aber nur eine bestimmte Kapazität. Ist diese erschöpft, wird der gesamte Speicherbereich zur Diskette übertragen. Dabei kann es vorkommen, daß zwischen der Übertragung ein Programmabbruch erfolgt und dadurch alle im Zwischenspeicher enthaltenen Daten verlorengehen. Der Befehl **FLUSH** umgeht die Zwischenspeicherung und überträgt alle ankommenden Daten sofort auf Diskette. FLUSH muß nach einem Schreibbefehl gesetzt werden.

Beispiel:

```
10 PRINT CHR$(4);"OPEN FILE"
20 PRINT CHR$(4);"WRITE FILE"
30 PRINT "ABCDEFG":PRINT CHR$(4);"FLUSH FILE"
40 PRINT CHR$(4);"CLOSE FILE"
50 END
```

Die Übertragung kann in ProDOS durch den Befehl MON unsichtbar gemacht werden. Er wird mit einem SYNTAX ERROR quittiert. NOMON wird akzeptiert, wird jedoch vom Computer nicht ausgeführt. Die Befehle DELETE, RENAME, APPEND, READ, WRITE, RUN, OPEN und CLOSE werden wie bei DOS benutzt. Mit dem Befehl CHAIN können Applesoft-Programme nachgeladen werden. Die Funktion POSITION wird in ProDOS nicht verlangt, kann aber verarbeitet werden. Das Kommando FRE löst eine Garbage Collection (Speicherbereinigung) aus. Die Ausführungen über ProDOS können nur unvollständig erfolgen, da bis zur Auflage dieses Buches keine weiteren Informationen vorlagen. Sie sollten auf jeden Fall bei einer intensiven Beschäftigung mit ProDOS weitere Literatur heranziehen.

# 10

# Der System-Monitor

## 10 Der System-Monitor

Wie im menschlichen Gehirn, so werden auch im Mikroprozessor Daten miteinander verglichen und eingeordnet. Wir nennen diese Vorgänge beim Menschen Erinnerungen. Man lernt und kann sich später wieder daran erinnern. Dieses "Wissen" ist in Form von festprogrammierten Befehlsrastern im Mikroprozessor Ihres Computers enthalten.

Da die CPU (Central Processor Unit, zu deutsch: Zentrale Rechner-Einheit), wie wir den Mikroprozessor auch nennen, nur Anweisungen in Form von Schaltzuständen versteht, brauchen wir eine Art Übersetzer.

Ein Bit ist die kleinste Einheit, die unser Computer versteht. Dieses Bit kann zwei Zustände haben: Aus oder Ein (0 oder 1). Dabei wird die Spannung nicht aus- und eingeschaltet, sondern es wird ein Spannungsunterschied erzeugt. Die Befehle wandern dabei parallel über einen Datenbus, der aus acht Leitungen besteht, zum Mikroprozessor. Jede dieser Leitungen kann die zwei Schaltzustände annehmen.

Natürlich ist es nicht ganz so einfach, wie es hier beschrieben wurde. Dennoch müßte das Prinzip klargeworden sein. Werden 8 Bit zusammengefaßt, so sprechen wir von einem Byte.

**Merken wir uns!**

* Ein Bit ist die kleinste Informationseinheit. Mit dem Wert eines Bits kann man einen Schaltzustand darstellen, z.B. ein/aus, ja/nein oder Spannung/keine Spannung.

* 8 Bit sind 1 Byte

Was sind nun aber 128 KByte ?

Das haben Sie bestimmt in Ihrer Apple-Dokumentation gelesen. Das K steht für Kilo, was soviel wie Tausend bedeutet. Aber es sind nicht genau 1000, sondern 1024 Byte, die einem KByte entsprechen, d.h., 128 KByte sind also 131072 Byte. Im Zusammenhang mit der Speicherkapazität des Apple IIc heißt das, daß Ihr Computer 128 x

1024 Byte Speicherzellen hat. Das sind 131072 Byte x 8 Bit = 1.048.576 Bit.

## 10.1 Die Bedeutung des Monitors

Ein großer Vorteil des Apple IIc ist sein eingebauter Monitor. Nicht der Monitor, auf dem wir unsere Bildschirmdarstellungen sehen, sondern der Monitor, der uns in die innersten Programme unseres Computers blicken läßt. Er dient dazu, einzelne Speicherzellen zu manipulieren oder ganze Programme über ihn einzugeben. Der Monitor ist ein Programm, das genauso wie der BASIC-Interpreter nur ganz bestimmte Anweisungen und Funktionen versteht.

### Hexadezimale Zahlen

In der Maschinenprogrammierung verwendet man das hexadezimale Zahlensystem. Dem dezimalen Zahlensystem liegen 10 Ziffern zugrunde.

0 1 2 3 4 5 6 7 8 9

Wird die zehnte Stelle erreicht, erscheint ein Übertrag auf die zweite Stelle.

10 11 12 13 14 15 ....

Dem hexadezimalen Zahlensystem dagegen liegen 16 Zeichen zugrunde.

0 1 2 3 4 5 6 7 8 9 A B C D E F

Wie Sie sehen können, gibt es bei der zehnten Stelle (dezimal gerechnet) keinen Übertrag. Dafür zählen wir hexadezimal mit A B C D E F weiter. Erst jetzt wird ein Übertrag auf die nächste Stelle vorgenommen.

8 9 A B C D E F 10 11 12 ....

In der dezimalen Schreibweise sprechen wir jetzt von der Zahl "Zwölf". In hexadezimaler Schreibweise ist das die Zahl "Eins Zwei". Um dezimale Werte von hexadezimalen Werten unterscheiden zu können, wird hexadezimalen Zahlen ein Dollarzeichen ($) vorangestellt. Im Monitor des Apple IIc brauchen wir das nicht zu tun, denn er denkt nur hexadezimal.

Der Mikroprozessor hat 8 Datenleitungen. Jede dieser Leitungen kann zwei Zustände einehmen: 1 oder 0. Bei zwei Datenleitungen haben wir schon vier Möglichkeiten, 00 01 10 11. Dieser Wert nimmt von Datenleitung zu Datenleitung sprunghaft zu. Da in unserem Computer ein 8-Bit-Mikroprozessor eingebaut ist, sehen wir uns einmal an, welche Möglichkeiten er uns bietet.

| | | | |
|---|---|---|---|
| 1 Datenleitung | = | 2 | Möglichkeiten |
| 2 Datenleitungen | = | 4 | Möglichkeiten |
| 3 Datenleitungen | = | 8 | Möglichkeiten |
| 4 Datenleitungen | = | 16 | Möglichkeiten |
| 5 Datenleitungen | = | 32 | Möglichkeiten |
| 6 Datenleitungen | = | 64 | Möglichkeiten |
| 7 Datenleitungen | = | 128 | Möglichkeiten |
| 8 Datenleitungen | = | 256 | Möglichkeiten |

Die Rechnung ist ganz einfach. Sie nehmen 8 Leitungen. Die erste bietet 2 mögliche Zustände an. Der Computer versteht 1 oder 0. Das machen wir jetzt 8mal.

$$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$$

Diese 256 Möglichkeiten werden dazu benutzt, die Befehle an den Mikroprozessor zu leiten. Darüber hinaus besitzt er noch 16 Adressierungsleitungen. Sie dienen dazu, die einzelnen Speicherstellen aufzurufen. Wir rechnen:

$$2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2 = 65536$$

Der Computer kann also 65536 Speicherstellen ansprechen. Das entspricht 64 KByte. Der Apple IIc hat aber 128 KByte. Die anderen 65536 Speicherstellen werden durch die Umschaltung des Speichers erreicht. Wird eine Speicherzelle oder ein ganzer Speicherblock benötigt, schaltet der Computer einfach auf diesen Block um und kann die Speicherstellen abfragen. Man spricht dabei von "Bankswitching".

## 10.2 Starten des System-Monitors

In Applesoft-BASIC und Integer-BASIC kann der System-Monitor aufgerufen werden. Dazu geben Sie folgenden Befehl ein:

CALL-151

oder

CALL 65385

CALL 65385 ist nur in Applesoft-BASIC möglich. In Integer-BASIC erfolgt die Fehlermeldung *** >32767 ERR. Jetzt zeigt der System-Monitor sein Bereitschaftszeichen, einen Stern (*), an. Sie können jetzt Monitoranweisungen eingeben. Diese dürfen aber nicht länger als 255 Zeichen sein. Um in den BASIC-Interpreter zurückkehren zu können, betätigen Sie bitte die CONTROL-Taste und die C-Taste gleichzeitig. Danach können Sie mit RETURN den System-Monitor wieder verlassen. Haben Sie DOS geladen, ist es möglich, mit der Befehlsfolge 3D0G und RETURN in den BASIC-Interpreter zurückkehren.

## 10.3 Speicheradressen anzeigen

Jeder Befehlscode ist in einer Speicherzelle im Computer abgelegt. Jede dieser Speicherstellen hat eine Adresse im gesamten Speicherbereich. Die Adressen werden durch vier hexadezimale Zahlen dargestellt und können von 0000 bis FFFF 64 KByte Speicherplatz ansprechen. Der Befehlscode, der in diesen Speicheradressen abgelegt ist, setzt sich aus zwei hexadezimalen Zeichen zusammen. Diese können einen Wert zwischen 00 und FF annehmen. Das sind 256 mögliche Befehlscodes. Der System-Monitor des Apple IIc muß also Daten und Adressen ansprechen können. Den Inhalt einer Speicheradresse können wir mit dem Monitor abrufen. Wir brauchen dazu nur die gewünschte Speicherstelle einzugeben.

Beispiel:

```
*300 (RETURN)

0300- 00
```

Nach der Eingabe der Zahl 300 wurde der Inhalt dieser Speicherstelle ausgegeben. In diesem Modus ist es möglich, den gesamten Speicherbereich anzuschauen. Es wäre aber sehr mühsam, jede Speicherstelle einzeln aufzurufen. Darum bietet der Monitor uns die Möglichkeit, auch mehrere Speicherstellen gleichzeitig auszugeben. Dazu geben wir die Anfangs- und Endadresse des Speicherbereiches ein, den wir anschauen wollen. Ist dieser größer, als der Bildschirm darstellen kann, so läßt der Monitor die Speicherdaten an uns wie in einem Film vorbeiziehen. Um aber den Überblick nicht zu verlieren, können wir das Listing mit der CONTROL- und S-Taste anhalten. Nach erneutem Betätigen der CONTROL- und S-Taste läuft das Listing weiter. Die Anfangs- und Endadresse müssen durch einen Punkt getrennt sein.

Beispiel: *100.150

```
0100- 37 30 00 30 30 30 30 30
0108- 30 30 42 BE B9 B7 B4 92
0110- 80 AE 80 AE 90 B3 36 70
0118- A7 82 D6 F4 D4 7C A1 32
0120- D5 43 D4 25 5C FB 68 FA
0128- A5 3A FB 65 FA A5 68 FA
0130- FF FF FF FF 00 00 00 00
0138- FF FF FF 17 26 FC BE 6F
0140- BE 6F 6F BE 66 BE C6 42
0148- BE B9 B7 B4 92 80 AE 80
0150- AE
```

## 10.4 Speicherinhalte verändern

Wir können im System-Monitor auch Speicherinhalte verändern. Dazu muß einem bestimmten Adreßwert ein Datenwert zugeordnet werden. Dieser kann im hexadezimalen Bereich zwischen 00 und FF liegen. Um z.B. den Code 18 in Speicherstelle 0300 ablegen zu können, ist folgende Eingabeweise einzuhalten:

*0300:18

Der System-Monitor legt den hexadezimalen Wert 18 in der Speicherstelle mit der Adresse 0300 ab. Die Eingabe erfolgt immer in hexadezimaler Form. Die Null vor 300 muß nicht unbedingt eingegeben werden. Der System-Monitor stellt der Zahl 300 automatisch eine Null vor, da es sich um eine Adressenangabe handelt. Sie können auch nur 300 eingeben und mit RETURN bestätigen. Der Monitor zeigt Ihnen die Adresse mit dem Speicherinhalt durch einen Trennstrich getrennt an. In der Zeile, in der das Bereitschafts-

zeichen steht, kann der neue Wert eingegeben werden. Dazu muß vor den neuen Wert ein Doppelpunkt (:) gesetzt werden. Wird die Eingabe mit RETURN abgeschlossen, so ersetzt der Monitor den angezeigten Wert mit dem eingegebenen.

```
*300 (RETURN)

0300- 00
*:18 (RETURN)

*300 (RETURN)

0300- 18
*
```

In diesem Beispiel ist eine Speicherstelle geändert worden. Um auch bei der Eingabe nicht jede Speicherzelle neu angeben zu müssen, kann nach der Startadresse eine Vielzahl von Daten gesetzt werden. Die Dateneingaben werden durch ein Leerzeichen getrennt. Der System-Monitor ordnet die Werte nach aufsteigenden Adressen.

Beispiel:

```
*300: FF FF FF FF FF FF FF FF FF FF
```

Den Adressen 0300 bis 0309 wurde der Wert FF zugeordnet. In einer Zeile können so maximal 255 Zeichen eingegeben werden. Wird mit RETURN die Eingabe bestätigt, können in der nächsten Zeile weitere Daten hinzugefügt werden. Sie müssen dazu nur einen Doppelpunkt nach dem Bereitschaftszeichen setzen. Der System-Monitor hat sich die letzte Speicheradresse gemerkt und übernimmt die folgende Eingabe in die Speicherstelle, die der letzten folgt.

Beispiel:

```
*300: 1 2 3 4 5 6 (RETURN)

*:7 8 9  (RETURN)

*300.308
```

```
0300- 01 02 03 04 05 06 07 08
0308- 09
```

## 10.5 Verschieben eines Speicherbereichs

Der System-Monitor kann Speicherbereiche verschieben. Dazu muß erst die Zieladresse angegeben werden. Nach einer Klammer (<) kommt die Anfangsadresse und durch einen Punkt getrennt die Endadresse des Bereichs, der verschoben werden soll. Der Buchstabe M für Move (bewegen) schließt die Befehlssequenz ab.

Beispiel:

```
*300:0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3

*0000.0017

0000- 00 00 00 00 00 00 00 00
0008- 00 00 00 00 00 00 00 00
0010- 00 00 00 00 00 00 00 00
*0300.0317

0300- 00 01 02 03 04 05 06 07
0308- 08 09 00 01 02 03 04 05
0310- 06 07 08 09 00 01 02 03
*0<300.317M

*0000.0017

0000- 00 01 02 03 04 05 06 07
0008- 08 09 00 01 02 03 04 05
0010- 06 07 08 09 00 01 02 03
*0300.0317

0300- 00 01 02 03 04 05 06 07
0308- 08 09 00 01 02 03 04 05
0310- 06 07 08 09 00 01 02 03
```

## 10.6 Vergleichen von zwei Speicherbereichen

Wir haben im letzten Beispiel einen Speicherbereich verschoben. Bei diesem kleinen Beispiel ist eine Überprüfung der Bereiche noch leicht zu bewerkstelligen. Sind jedoch größere Speicherbereiche miteinander zu vergleichen, kann uns der System-Monitor helfen. Wir wollen aber beim letzten Beispiel bleiben und den Speicherbereich 0300 bis 0317 mit dem von 0000 bis 0017 vergleichen. Dazu verwenden wir das gleiche Eingabeformat wie schon bei der Move-Anweisung. Statt des Buchstabens M wird jetzt der Buchstabe V eingesetzt. Er steht für das englische Wort Verify. Dabei kann $0300 oder $0000 als Zieladresse angegeben werden.

Beispiel:

```
*0000<0300.0317V

*0:22 33 44

*0000<0300.0317V

0300-00 (22)
0301-01 (33)
0302-02 (44)
```

In der ersten Zeile des ausgedruckten Protokolls wurden keine Fehler ausgegeben. In der zweiten Zeile wurden die Speicherstellen 0000, 0001 und 0002 gezielt manipuliert. Nach erneutem Verify hat der System-Monitor die Fehler ausgegeben. Sie sollten vermeiden, daß sich die vergleichenden Bereiche überschneiden, da ansonsten der System-Monitor außer Kontrolle geraten kann.

## 10.7 Starten von Monitorprogrammen

Wenn Sie ein Programm im Monitor erstellt haben, kann es auch von dort gestartet werden. Dazu wird die Startadresse des Programms eingegeben und mit dem Buchstaben G abgeschlossen. So können auch Unterprogramme, die im ROM (Festwertspeicher) des Computers liegen, aufgerufen werden.

Beispiel:

```
*FB60G
```

Wird das Unterprogramm ab der Speicherstelle FB60 gestartet, so löscht der Computer den Bildschirm und schreibt in die oberste Zeile "Apple IIc". Wollen Sie sich das Programm anschauen, geben Sie hinter die Startadresse den Buchstaben L ein. Dieser Befehl disassembliert ein Maschinenprogramm. Das heißt, die hexadezimalen Werte in den Speicheradressen werden in lesbare Mnemonics umgewandelt. So können die Befehlslängen festgestellt werden. Im Anhang G werden die einzelnen Mnemonics genauer beschrieben.

## 10.8 Rechnen im System-Monitor

Im System-Monitor können einfache hexadezimale Additionen und Subtraktionen ausgeführt werden. Dazu müssen die Zahlen durch ein Plus- oder Minuszeichen getrennt werden.

Beispiel:

```
*12+12
=24
*FF-23
=DC
*45+F
=54
```

Wie in BASIC, so kann auch im System-Monitor die Ausgabe invers erfolgen. Dies erfolgt durch I für Inverse. Soll in den Normalzustand geschaltet werden, so ist der Buchstabe N einzugeben.

Die Registerinhalte des 65C02-Registers können mit der Tastenfolge CTRL-E und RETURN ausgegeben werden.

Im Anhang H ist das gesamte Listing des Monitor-ROMS ab der Speicherstelle $F800 bis FFFF abgedruckt. In diesem Bereich sind die Monitorunterroutinen abgelegt.

# Anhang

## A: Reservierte Worte des Applesoft-Interpreters

ABS
AND
ASC
AT
ATN
CALL
CHR$
CLEAR
COLOR=
CONT
COS
DATA
DEF
DEL
DIM
DRAW
END
EXP
FLASH
FN
FOR
FRE
GET
GOSUB
GOTO
GR
HCOLOR=
HGR
HGR2
HIMEM:
HLIN
HOME
HPLOT
HTAB
IF
IN#
INPUT
INT
INVERSE
LEFT$
LEN
LET
LIST
LOAD
LOG
LOMEM:
MID$
NEW
NEXT
NORMAL
NOT
NOTRACE
ON
ONERR
OR
PDL
PEEK
PLOT
POKE
POP
POS
PRINT
PR#
READ
RECALL
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
ROT=
RUN
SAVE
SCALE=
SCRN(
SGN
SHLOAD
SIN
SPC(
SPEED=
SQR
STEP
STOP
STORE
STR$
TAB(
TAN
TEXT
THEN
TO
TRACE
USR
VAL
VLIN
VTAB
WAIT
XDRAW

## B: Fehlermeldungen in Applesoft-BASIC

**?BAD SUBSCRIPT ERROR**

Der definierte Bereich einer dimensionierten Variablen wurde unter- oder überschritten, bzw. die Indexpositionen sind größer als ihr Höchstwert.

**?CAN'T CONTINUE ERROR**

Mit dem CONT-Befehl wurde versucht, ein nichtvorhandenes, abgeändertes oder fehlerhaftes Programm zu starten.

**?DIVISION BY ZERO ERROR**

Sie haben durch Null dividiert! Dies ist nicht möglich.

**?FORMULA TOO COMPLEX ERROR**

Es wurde versucht, nach einer IF-THEN-Befehlsfolge eine weitere einzusetzen.

**?ILLEGAL DIRECT ERROR**

Die Anweisungen DATA, DEF FN, GET, INPUT, ON ERR GOTO, RESUME wurden in der Betriebsart "Direktanweisung" benutzt. Diese Befehle dürfen aber nur in der Betriebsart "Programmieren" verwendet werden.

**?ILLEGAL QUANTITY ERROR**

Ein Parameterwert liegt außerhalb des definierten Bereiches.

**?NEXT WITHOUT FOR ERROR**

Eine FOR-Anweisung wurde nicht eingegeben.

**?OUT OF DATA ERROR**

Der Wert einer definierten Konstanten wurde überschritten.

**?OUT OF MEMORY ERROR**

Die Kapazität des Speichers wurde überschritten. Es wurden zu viele Variablen definiert oder zu viele Unterprogramme ineinandergeschachtelt.

**?OVERFLOW ERROR**

Eine Zahl zwischen -1.7E+38 und 1.7E+38 wurde versucht zu berechnen oder einzugeben.

**?REDIM'D ARRAY ERROR**

Es darf eine dimensionierte Variable nur einmal dimensioniert werden.

**?RETURN WITHOUT GOSUB ERROR**

Es wurden mehr Unterprogramme mit RETURN abgeschlossen, als GOSUB-Anweisungen vorhanden sind.

**?STRING TOO LONG ERROR**

In einer Programmzeile sind mehr als 255 Zeichen.

**?SYNTAX ERROR**

Eine Anweisung wurde falsch eingegeben. Diese Fehlermeldung tritt auch dann auf, wenn ein Fehler auftaucht, für den es keine Fehlermeldung gibt.

**?TYPE MISMATCH ERROR**

Es wurde versucht, einer Textvariablen einen numerischen Ausdruck oder einer numerischen Variablen einen Text zuzuordnen.

**?UNDEFINED FUNCTION ERROR**

Die aufgerufene Funktion ist fehlerhaft.

**?UNDEFINED STATEMENT ERROR**

Es wurde eine Programmzeilennummmer als Verzweigungsadresse angegeben, die nicht existiert.

**Fehlermeldungen in Integer-BASIC**

***** >255 ERR**

Der zulässige Zahlenbereich zwischen 0 und 255 wurde überschritten.

***** >32767 ERR**

Der zulässige Zahlenbereich zwischen -32767 und 32767 wurde überschritten.

***** 16 FORS ERR**

Es wurde versucht, mehr als 16 FOR..NEXT-Schleifen zu beginnen.

***** 16 GOSUB ERR**

Die Anzahl der ausgeführten GOSUB-Anweisungen ist um 17 größer als die Anzahl der ausgeführten RETURN-Anweisungen.

***** BAD BRANCH ERR**

Eine Programmzeile wurde in einer Verzweigung angegeben, die nicht existiert.

***** BAD NEXT ERR**

Eine aufgerufene FOR-Anweisung konnte nicht gefunden werden.

***** BAD RETURN ERR**

Es wurden mehr Unterprogramme mit RETURN abgeschlossen, als GOSUB-Anweisungen vorhanden sind.

***** DIM ERR**

Es darf eine dimensionierte Variable nur einmal dimensioniert werden.

***** MEM FULL ERR**

Die Kapazität des Speichers wurde überschritten. Es wurden zu viele Variablen definiert oder zu viele Unterprogramme ineinandergeschachtelt.

***** NO END ERR**

In Integer-BASIC muß ein Programm immer mit der Anweisung END abgeschlossen werden. Diese Anweisung fehlt.

***** RANGE ERR**

Eine indizierte Variable liegt außerhalb des zulässigen Bereichs, oder der Wert des Arguments einer Anweisung wurde überschritten.

***** RETYPE ERR**

Eine INPUT-Eingabe ist falsch. Die Eingabe enthält den falschen Datentyp.

***** STRING ERR**

Eine Textvariable wurde falsch behandelt.

***** STRING OVFL ERR**

Einer Textvariablen wurde ein längerer Text zugeordnet, als ihre Dimensionierung erlaubt.

***** SYNTAX ERR**

Die abgearbeitete Programmzeile enthält eine fehlerhafte Anweisung, oder es wurde versucht, eine syntaktisch falsche Eingabe zu machen.

***** TOO LONG ERR**

Eine Programmzeile enthält mehr als 128 Zeichen.

## C: Maschinenunterprogramme

Der Befehl CALL in BASIC läßt den Computer zu einem Maschinenunterprogramm verzweigen. Ist das Unterprogramm ausgeführt, kehrt die Routine wieder zum BASIC-Programm zurück. Das BASIC-Programm bleibt dabei im Arbeitsspeicher erhalten.

| | |
|---|---|
| CALL -151 | bringt Sie in den System-Monitor |
| CALL -198 | erzeugt ein Tonsignal |
| CALL -211 | erzeugt eine ERR-Meldung und ein Tonsignal |
| CALL -415 | ist die Einsprungadresse des Disassemblers, in der Speicherstelle dezimal 58 und 59 muß die Startadresse stehen |
| CALL -678 | wartet auf die Eingabe von RETURN |
| CALL -756 | wartet auf eine Tastenbetätigung |
| CALL -868 | löscht den Bildschirm von der gegenwärtigen Cursorposition bis zum Zeilenende |
| CALL -912 | rollt den Text auf dem Bildschirm um eine Zeile nach oben |
| CALL -922 | bringt eine Leerzeile auf den Textbildschirm |
| CALL -936 | hat die gleiche Funktion wie die Anweisung HOME |
| CALL -958 | löscht den Bildschirm bis zum Ende der Seite |
| CALL -1184 | löscht den Bildschirm und gibt "Apple IIc" aus |
| CALL -1321 | gibt die Registerinhalte des 65C02 aus |
| CALL -1370 | startet das Diskettenlaufwerk von neuem |
| CALL -1438 | erzeugt einen Pseudo-System-Reset |

CALL -1994 schreibt die Textseite 1 mit Zeichen voll. Befindet sich der Apple im GR-Modus, wird der 40 mal 40 große Grafikbildschirm schwarz.

CALL -1998 schreibt die Textseite 1 mit Zeichen voll. Im GR-Modus wird der 40 mal 40 große Grafikbildschirm schwarz, und die vier Zeilen Text werden mit inversen Zeichen gefüllt.

CALL -3086 löscht den Grafikbildschirm

## D: ASCII-Code der deutschen Tastenbelegung

| ASCII-Code | Bedeutung | Taste(n) | ASCII-Code | Bedeutung | Taste(n) |
|---|---|---|---|---|---|
| 0 | | CTRL - § | 32 | Space | Leer- |
| 1 | | CTRL - A | | | zeichen |
| 2 | | CTRL - B | 33 | ! | ! |
| 3 | | CTRL - C | 34 | " | " |
| 4 | | CTRL - D | 35 | # | # |
| 5 | | CTRL - E | 36 | $ | $ |
| 6 | | CTRL - F | 37 | % | % |
| 7 | Glocke | CTRL - G | 38 | & | & |
| 8 | 1 Stelle | CTRL - H | 39 | ' | ' |
| | zurück | | 40 | ( | ( |
| 9 | | CTRL - I | 41 | ) | ) |
| 10 | Zeilen- | CTRL - J | 42 | * | * |
| | vorschub | | 43 | + | + |
| 11 | | CTRL - K | 44 | , | , |
| 12 | | CTRL - L | 45 | - | - |
| 13 | Wagen- | CTRL - M | 46 | . | . |
| | rücklauf | | 47 | / | / |
| 14 | | CTRL - N | 48 | 0 | 0 |
| 15 | | CTRL - O | 49 | 1 | 1 |
| 16 | | CTRL - P | 50 | 2 | 2 |
| 17 | | CTRL - Q | 51 | 3 | 3 |
| 18 | | CTRL - R | 52 | 4 | 4 |
| 19 | | CTRL - S | 53 | 5 | 5 |
| 20 | | CTRL - T | 54 | 6 | 6 |
| 21 | Tabulator | CTRL - U | 55 | 7 | 7 |
| 22 | | CTRL - V | 56 | 8 | 8 |
| 23 | | CTRL - W | 57 | 9 | 9 |
| 24 | Zeile | CTRL - X | 58 | : | : |
| | löschen | | 59 | ; | ; |
| 25 | | CTRL - Y | 60 | < | < |
| 26 | | CTRL - Z | 61 | = | = |
| 27 | | Esc | 62 | > | > |
| 28 | | | 63 | ? | ? |
| 29 | | CTRL- ^ | 64 | § | § |
| | | Shift-M | 65 | A | A |
| 30 | | CTRL - | 66 | B | B |
| 31 | | | 67 | C | C |

**ASCII-Code der deutschen Tastenbelegung Teil 2**

| ASCII-Code | Bedeutung | Taste(n) | ASCII-Code | Bedeutung | Taste(n) |
|---|---|---|---|---|---|
| 68 | D | D | 101 | e | e |
| 69 | E | E | 102 | f | f |
| 70 | F | F | 103 | g | g |
| 71 | G | G | 104 | h | h |
| 72 | H | H | 105 | i | i |
| 73 | I | I | 106 | j | j |
| 74 | J | J | 107 | k | k |
| 75 | K | K | 108 | l | l |
| 76 | L | L | 109 | m | m |
| 77 | M | M | 110 | n | n |
| 78 | N | N | 111 | o | o |
| 79 | O | O | 112 | p | p |
| 80 | P | P | 113 | q | q |
| 81 | Q | Q | 114 | r | r |
| 82 | R | R | 115 | s | s |
| 83 | S | S | 116 | t | t |
| 84 | T | T | 117 | u | u |
| 85 | U | U | 118 | v | v |
| 86 | V | V | 119 | w | w |
| 87 | W | W | 120 | x | x |
| 88 | X | X | 121 | y | y |
| 89 | Y | Y | 122 | z | z |
| 90 | Z | Z | 123 | ä | ä |
| 91 | Ä | Ä | 124 | ö | ö |
| 92 | Ö | Ö | 125 | ü | ü |
| 93 | Ü | Ü | 126 | ß | ß |
| 94 | ^ | ^ | 127 | | |
| 95 | | | 128 | | |
| 96 | ` | ` | 129 | | |
| 97 | a | a | 130 | | |
| 98 | b | b | 131 | | |
| 99 | c | c | | | |
| 100 | d | d | | | |

**ASCII-Code der US-Tastenbelegung**

| ASCII-Code | Bedeutung | Taste(n) | ASCII-Code | Bedeutung | Taste(n) |
|---|---|---|---|---|---|
| 0 | | CTRL - | 32 | Space | Leer- |
| 1 | | CTRL - A | | | zeichen |
| 2 | | CTRL - B | 33 | ! | ! |
| 3 | | CTRL - C | 34 | " | " |
| 4 | | CTRL - D | 35 | # | # |
| 5 | | CTRL - E | 36 | $ | $ |
| 6 | | CTRL - F | 37 | % | % |
| 7 | Glocke | CTRL - G | 38 | & | & |
| 8 | 1 Stelle | CTRL - H | 39 | ' | ' |
| | zurück | | 40 | ( | ( |
| 9 | | CTRL - I | 41 | ) | ) |
| 10 | Zeilen- | CTRL - J | 42 | * | * |
| | vorschub | | 43 | + | + |
| 11 | | CTRL - K | 44 | , | , |
| 12 | | CTRL - L | 45 | - | - |
| 13 | Wagen- | CTRL - M | 46 | . | . |
| | rücklauf | | 47 | / | / |
| 14 | | CTRL - N | 48 | 0 | 0 |
| 15 | | CTRL - O | 49 | 1 | 1 |
| 16 | | CTRL - P | 50 | 2 | 2 |
| 17 | | CTRL - Q | 51 | 3 | 3 |
| 18 | | CTRL - R | 52 | 4 | 4 |
| 19 | | CTRL - S | 53 | 5 | 5 |
| 20 | | CTRL - T | 54 | 6 | 6 |
| 21 | Tabulator | CTRL - U | 55 | 7 | 7 |
| 22 | | CTRL - V | 56 | 8 | 8 |
| 23 | | CTRL - W | 57 | 9 | 9 |
| 24 | Zeile | CTRL - X | 58 | : | : |
| | löschen | | 59 | ; | ; |
| 25 | | CTRL - Y | 60 | < | < |
| 26 | | CTRL - Z | 61 | = | = |
| 27 | | Esc | 62 | > | > |
| 28 | | | 63 | ? | ? |
| 29 | | CTRL- ^ | 64 | § | § |
| | | Shift-M | 65 | A | A |
| 30 | | CTRL - | 66 | B | B |
| 31 | | | 67 | C | C |

**ASCII-Code der US-Tastenbelegung Teil 2**

| ASCII-Code | Bedeutung | Taste(n) | ASCII-Code | Bedeutung | Taste(n) |
|---|---|---|---|---|---|
| 68 | D | D | 101 | e | e |
| 69 | E | E | 102 | f | f |
| 70 | F | F | 103 | g | g |
| 71 | G | G | 104 | h | h |
| 72 | H | H | 105 | i | i |
| 73 | I | I | 106 | j | j |
| 74 | J | J | 107 | k | k |
| 75 | K | K | 108 | l | l |
| 76 | L | L | 109 | m | m |
| 77 | M | M | 110 | n | n |
| 78 | N | N | 111 | o | o |
| 79 | O | O | 112 | p | p |
| 80 | P | P | 113 | q | q |
| 81 | Q | Q | 114 | r | r |
| 82 | R | R | 115 | s | s |
| 83 | S | S | 116 | t | t |
| 84 | T | T | 117 | u | u |
| 85 | U | U | 118 | v | v |
| 86 | V | V | 119 | w | w |
| 87 | W | W | 120 | x | x |
| 88 | X | X | 121 | y | y |
| 89 | Y | Y | 122 | z | z |
| 90 | Z | Z | 123 | | |
| 91 | [ | [ | 124 | | |
| 92 | \ | \ | 125 | | |
| 93 | ] | ] | 126 | | |
| 94 | ^ | ^ | 127 | | |
| 95 | _ | _ | 128 | | |
| 96 | ` | ` | 129 | | |
| 97 | a | a | 130 | | |
| 98 | b | b | 131 | | |
| 99 | c | c | | | |
| 100 | d | d | | | |

## F: OP-CODE der CPU 65C02

| Mnemonics + Beschreibung | Adressie-rungsart | OP-Code (hex) | erford.Takt-impulse | belegte Bytes |
|---|---|---|---|---|
| **ADC** | IMM | 69 | 2 | 2 |
| Addiere den | ABS | 6D | 4 | 3 |
| Inhalt des | Z | 65 | 3 | 2 |
| Speichers | (IND,X) | 61 | 6 | 2 |
| zum Akku mit | (IND),Y | 71 | 5 | 2 |
| Carry | Z,X | 75 | 4 | 2 |
| | ABS,X | 7D | 4 | 3 |
| | ABS,Y | 79 | 4 | 3 |
| | IND | 72 | 5 | 2 |
| **AND** | IMM | 29 | 2 | 2 |
| "AND" Speicher | ABS | 2D | 4 | 3 |
| mit Akku | Z | 25 | 3 | 2 |
| | (IND,X) | 21 | 6 | 2 |
| | (IND),Y | 31 | 5 | 2 |
| | Z,X | 35 | 4 | 2 |
| | ABS,X | 3D | 4 | 3 |
| | ABS,Y | 39 | 4 | 3 |
| | IND | 32 | 5 | 2 |
| **ASL** | Accu | 0A | | 1 |
| Akku oder | ABS | 0E | 6 | 3 |
| Speicher | Z | 06 | 5 | 2 |
| ein Bit | A | 0A | 2 | 1 |
| nach links | Z,X | 16 | 6 | 2 |
| schieben | ABS,X | 1E | 7 | 3 |
| **BCC** | REL | 90 | 2/3 | 2 |
| Verzweigung bei gelöschtem Übertrag(Carry) | | | | |
| **BCS** | REL | B0 | 2/3 | 2 |
| Verzweigung bei gesetztem Übertrag(Carry) | | | | |

| Mnemonics + Beschreibung | Adressie-rungsart | OP-Code (hex) | erford.Takt-impulse | belegte Bytes |
|---|---|---|---|---|
| **BEQ** Verzweigung bei Null | REL | F0 | 2/3 | 2 |
| **BIT** Vergleiche Bits aus Speicher mit Akku | ABS | 2C | 4 | 3 |
| | Z | 24 | 3 | 2 |
| | IM | 89 | 2 | 2 |
| | Z,X | 34 | 4 | 2 |
| | ABS,X | 3C | 4 | 3 |
| **BMI** Verzweige bei Minus | REL | 30 | 2/3 | 2 |
| **BNE** Verzweige bei ungleich Null | REL | D0 | 2/3 | 2 |
| **BPL** Verzweige bei Plus | REL | 10 | 2/3 | 2 |
| **BRA** Verzweige immer relativ | REL | 80 | 2 | 2 |
| **BRK** Interrupt erzwingen | IMP | 00 | 7 | 1 |
| **BVC** Verzweigung bei gelöschtem Überlauf(Ovfl.) | REL | 50 | 2/3 | 2 |
| **BVS** Verzweigung bei gesetztem Überlauf(Ovfl.) | REL | 70 | 2/3 | 2 |

| Mnemonics + Beschreibung | Adressierungsart | OP-Code (hex) | erford.Taktimpulse | belegte Bytes |
|---|---|---|---|---|
| **CLC** Lösche Übertrag /Carry) | IMP | 18 | 2 | 1 |
| **CLD** Lösche Dezimalmodus | IMP | D8 | 2 | 1 |
| **CLI** Lösche Interruptsperre | IMP | 58 | 2 | 1 |
| **CLV** Lösche Überlauf(Ovfl.) | IMP | B8 | 2 | 1 |
| **CMP** Vergleiche Akku mit Speicher | IMM | C9 | 2 | 2 |
| | ABS | CD | 4 | 3 |
| | Z | C5 | 3 | 2 |
| | (IND,X) | C1 | 6 | 2 |
| | (IND),Y | D1 | 5 | 2 |
| | Z,X | D5 | 4 | 2 |
| | ABS,X | DD | 4 | 3 |
| | ABS,Y | D9 | 4 | 3 |
| | IND | D2 | 5 | 2 |
| **CPX** Vergleiche Indexregister X mit Speicher | IMM | E0 | 2 | 2 |
| | ABS | EC | 4 | 3 |
| | Z | E4 | 3 | 2 |
| **CPY** Vergleiche Indexregister Y mit Speicher | IMM | C0 | 2 | 2 |
| | ABS | CC | 4 | 3 |
| | Z | C4 | 3 | 2 |

| Mnemonics + Beschreibung | Adressierungsart | OP-Code (hex) | erford.Taktimpulse | belegte Bytes |
|---|---|---|---|---|
| **DEC** | ABS | CE | 6 | 3 |
| Vermindere | Z | C6 | 5 | 2 |
| um 1 | Z,X | D6 | 6 | 2 |
| | ABS,X | DE | 7 | 3 |
| | IMP | 3A | 2 | 1 |
| **DEX** | IMP | CA | 2 | 1 |
| Vermindere Indexregister X um 1 | | | | |
| **DEY** | IMP | 88 | 2 | 1 |
| Vermindere Indexregister Y um 1 | | | | |
| **EOR** | IMM | 49 | 2 | 2 |
| Logisches | ABS | 4D | 4 | 3 |
| exklusives | Z | 45 | 3 | 2 |
| ODER: | (IND,X) | 41 | 6 | 2 |
| Speicher | (IND),Y | 51 | 5 | 2 |
| mit Akku | Z,X | 55 | 4 | 2 |
| | ABS,X | 5D | 4 | 3 |
| | ABS,Y | 59 | 4 | 3 |
| | IND | 52 | 5 | 2 |
| **INC** | ABS | EE | 6 | 3 |
| Erhöhe | Z | E6 | 5 | 2 |
| Speicher | Z,X | F6 | 6 | 2 |
| um 1 | ABS,X | FE | 7 | 3 |
| | IMP | 1A | 2 | 1 |
| **INX** | IMP | E8 | 2 | 1 |
| Erhöhe Indexregister X um 1 | | | | |

| Mnemonics + Beschreibung | Adressie-rungsart | OP-Code (hex) | erford.Takt-impulse | belegte Bytes |
|---|---|---|---|---|
| **INY** Erhöhe Indexregister Y um 1 | IMP | C8 | 2 | 1 |
| **JMP** Unbedingter Sprung | ABS | 4C | 3 | 3 |
| | IND | 6C | 5 | 3 |
| | (IND),Y | 7C | 6 | 3 |
| **JSR** Springe ins Unterprogramm | ABS | 20 | 6 | 3 |
| **LDA** Lade Speicher in den Akku | IMM | A9 | 2 | 2 |
| | ABS | AD | 4 | 3 |
| | Z | A5 | 3 | 2 |
| | (IND,X) | A1 | 6 | 2 |
| | (IND),Y | B1 | 5 | 2 |
| | Z,X | B5 | 4 | 2 |
| | ABS,X | BD | 4 | 3 |
| | ABS,Y | B9 | 4 | 3 |
| | IND | B2 | 5 | 2 |
| **LDX** Lade Speicher in das Indexregister X | IMM | A2 | 2 | 2 |
| | ABS | AE | 4 | 3 |
| | Z | A6 | 3 | 2 |
| | Z,Y | B6 | 4 | 2 |
| | ABS,Y | BE | 4 | 3 |
| **LDY** Lade Speicher in das Indexregister Y | IMM | A0 | 2 | 2 |
| | ABS | AC | 4 | 3 |
| | Z | A4 | 3 | 2 |
| | Z,X | B4 | 4 | 2 |
| | ABS,X | BC | 4 | 3 |

| Mnemonics + Beschreibung | Adressierungsart | OP-Code (hex) | erford.Taktimpulse | belegte Bytes |
|---|---|---|---|---|
| **LSR** | ABS | 4E | 6 | 3 |
| (Akku oder | Z | 46 | 5 | 2 |
| Speicher) ein | A | 4A | 2 | 1 |
| Bit nach | Z,X | 56 | 6 | 2 |
| rechts | ABS,X | 5E | 7 | 3 |
| schieben | | | | |
| **NOP** | IMP | EA | 2 | 1 |
| Keine Operation | | | | |
| **ORA** | IMM | 09 | 2 | 2 |
| Logisch ODER: | ABS | 0D | 4 | 3 |
| Speicher mit | Z | 05 | 3 | 2 |
| Akku | (IND,X) | 01 | 6 | 2 |
| | (IND),Y | 11 | 5 | 2 |
| | Z,X | 15 | 4 | 2 |
| | ABS,X | 1D | 4 | 3 |
| | ABS,Y | 19 | 4 | 3 |
| | IND | 12 | 5 | 2 |
| **PHA** | IMP | 48 | 3 | 1 |
| Bringe Akku in das Stapelregister | | | | |
| **PHP** | IMP | 08 | 3 | 1 |
| Bringe den Prozessorstatus in das Stapelregister | | | | |
| **PHX** | IMP | DA | 3 | 1 |
| Bringe das Indexregister X in das Stapelregister | | | | |

| Mnemonics + Beschreibung | Adressierungsart | OP-Code (hex) | erford.Taktimpulse | belegte Bytes |
|---|---|---|---|---|
| **PHY** Bringe das Indexregister Y in das Stapelregister | IMP | 5A | 3 | 1 |
| **PLA** Hole Akku aus dem Stapelregister | IMP | 68 | 4 | 1 |
| **PLP** Hole den Prozessorstatus aus dem Stapelregister | IMP | 28 | 4 | 1 |
| **PLX** Hole das Indexregister X aus dem Statusregister | IMP | FA | 4 | 1 |
| **PLY** Hole das Indexregister Y aus dem Statusregister | IMP | 7A | 4 | 1 |
| **ROL** | ABS | 2E | 6 | 3 |
| Akku oder | Z | 26 | 5 | 2 |
| Speicher | Z,X | 36 | 6 | 2 |
| im Kreis nach | ABS,X | 3E | 7 | 3 |
| links schieben | A | 2A | 2 | 1 |

| Mnemonics + Beschreibung | Adressierungsart | OP-Code (hex) | erford.Taktimpulse | belegte Bytes |
|---|---|---|---|---|
| **ROR** | ABS | 6E | 6 | 3 |
| Akku oder | Z | 66 | 5 | 2 |
| Speicher | A | 6A | 2 | 1 |
| im Kreis nach | Z,X | 76 | 6 | 2 |
| rechts | ABS,X | 7E | 7 | 3 |
| schieben | | | | |
| **RTI** | IMP | 40 | 6 | 1 |
| Rücksprung aus dem Interrupt | | | | |
| **RTS** | IMP | 60 | 6 | 1 |
| Rücksprung aus einem Unterprogramm | | | | |
| **SBC** | IMM | E9 | 2 | 2 |
| Subtrahiere | ABS | ED | 4 | 3 |
| Speicher und | Z | E5 | 3 | 2 |
| Übertrag | (IND,X) | E1 | 6 | 2 |
| vom Akku | (IND),Y | F1 | 5 | 2 |
| | Z,X | F5 | 4 | 2 |
| | ABS,X | FD | 4 | 3 |
| | ABS,Y | F9 | 4 | 3 |
| | IND | F2 | 5 | 2 |
| **SEC** | IMP | 38 | 2 | 1 |
| Setze Übertrag (Carry) | | | | |
| **SED** | IMP | F8 | 2 | 1 |
| Setze Dezimalmodus | | | | |
| **SEI** | IMP | 78 | 2 | 1 |
| Setze Interrupt-sperre | | | | |

| Mnemonics + Beschreibung | Adressie-rungsart | OP-Code (hex) | erford.Takt-impulse | belegte Bytes |
|---|---|---|---|---|
| **STA** | ABS | 8D | 4 | 3 |
| Speichere aus | Z | 85 | 3 | 2 |
| dem Akku | (IND,X) | 81 | 6 | 2 |
| | (IND),Y | 91 | 6 | 2 |
| | Z,X | 95 | 4 | 2 |
| | ABS,X | 9D | 5 | 3 |
| | ABS,Y | 99 | 5 | 3 |
| | IND | 92 | 5 | 2 |
| **STX** | ABS | BE | 4 | 3 |
| Speichere | Z | 86 | 3 | 2 |
| aus dem | Z,Y | 96 | 4 | 2 |
| Indexreg. X | | | | |
| **STY** | ABS | 8C | 4 | 3 |
| Speichere | Z | 84 | 3 | 2 |
| aus dem | Z,X | 94 | 4 | 2 |
| Indexregister | | | | |
| Y | | | | |
| **STZ** | ABS | 9C | 4 | 3 |
| Setze Speicher | Z | 64 | 3 | 2 |
| auf Null | Z,X | 74 | 4 | 2 |
| | ABS,X | 9E | 5 | 3 |
| **TAX** | IMP | AA | 2 | 1 |
| Transportiere | | | | |
| Akku zum | | | | |
| Indexregister X | | | | |
| **TAY** | IMP | A8 | 2 | 1 |
| Transportiere | | | | |
| Akku zum | | | | |
| Indexregister Y | | | | |
| **TRB** | ABS | 1C | 6 | 3 |
| Test and reset | Z | 14 | 5 | 2 |
| memory bits | | | | |
| with accumulator | | | | |

| Mnemonics + Beschreibung | Adressie-rungsart | OP-Code (hex) | erford.Takt-impulse | belegte Bytes |
|---|---|---|---|---|
| **TSB** Test and set memory bits with accumulator | ABS<br>Z | 0C<br>04 | 6<br>5 | 3<br>2 |
| **TSX** Transportiere Statuszeiger zum Index-register X | IMP | BA | 2 | 1 |
| **TXA** Transportiere Indexregister X zu Akku | IMP | 8A | 2 | 1 |
| **TXS** Transportiere Indexregister X zum Status-zeiger | IMP | 9A | 2 | 1 |
| **TYS** Transportiere Indexregister Y zum Akku | IMP | 98 | 2 | 1 |

**F: Assemblerlisting des $F800-ROM**

```
F800-   4A          LSR
F801-   08          PHP
F802-   20 47 F8    JSR   $F847
F805-   28          PLP
F806-   A9 0F       LDA   #$0F
F808-   90 02       BCC   $F80C
F80A-   69 E0       ADC   #$E0
F80C-   85 2E       STA   $2E
F80E-   B1 26       LDA   ($26),Y
F810-   45 30       EOR   $30
F812-   25 2E       AND   $2E
F814-   51 26       EOR   ($26),Y
F816-   91 26       STA   ($26),Y
F818-   60          RTS
F819-   20 00 F8    JSR   $F800
F81C-   C4 2C       CPY   $2C
F81E-   B0 11       BCS   $F831
F820-   C8          INY
F821-   20 0E F8    JSR   $F80E
F824-   90 F6       BCC   $F81C
F826-   69 01       ADC   #$01
F828-   48          PHA
F829-   20 00 F8    JSR   $F800
F82C-   68          PLA
F82D-   C5 2D       CMP   $2D
F82F-   90 F5       BCC   $F826
F831-   60          RTS
F832-   A0 2F       LDY   #$2F
F834-   D0 02       BNE   $F838
F836-   A0 27       LDY   #$27
F838-   84 2D       STY   $2D
F83A-   A0 27       LDY   #$27
F83C-   A9 00       LDA   #$00
F83E-   85 30       STA   $30
F840-   20 28 F8    JSR   $F828
F843-   88          DEY
F844-   10 F6       BPL   $F83C
F846-   60          RTS
F847-   48          PHA
F848-   4A          LSR
```

```
F849-   29 03      AND   #$03
F84B-   09 04      ORA   #$04
F84D-   85 27      STA   $27
F84F-   68         PLA
F850-   29 18      AND   #$18
F852-   90 02      BCC   $F856
F854-   69 7F      ADC   #$7F
F856-   85 26      STA   $26
F858-   0A         ASL
F859-   0A         ASL
F85A-   05 26      ORA   $26
F85C-   85 26      STA   $26
F85E-   60         RTS
F85F-   A5 30      LDA   $30
F861-   18         CLC
F862-   69 03      ADC   #$03
F864-   29 0F      AND   #$0F
F866-   85 30      STA   $30
F868-   0A         ASL
F869-   0A         ASL
F86A-   0A         ASL
F86B-   0A         ASL
F86C-   05 30      ORA   $30
F86E-   85 30      STA   $30
F870-   60         RTS
F871-   4A         LSR
F872-   08         PHP
F873-   20 47 F8   JSR   $F847
F876-   B1 26      LDA   ($26),Y
F878-   28         PLP
F879-   90 04      BCC   $F87F
F87B-   4A         LSR
F87C-   4A         LSR
F87D-   4A         LSR
F87E-   4A         LSR
F87F-   29 0F      AND   #$0F
F881-   60         RTS
F882-   A6 3A      LDX   $3A
F884-   A4 3B      LDY   $3B
F886-   20 96 FD   JSR   $FD96
```

```
F889-   20 48 F9    JSR   $F948
F88C-   A1 3A       LDA   ($3A,X)
F88E-   A8          TAY
F88F-   4A          LSR
F890-   90 05       BCC   $F897
F892-   6A          ROR
F893-   B0 0C       BCS   $F8A1
F895-   29 87       AND   #$87
F897-   4A          LSR
F898-   AA          TAX
F899-   BD 62 F9    LDA   $F962,X
F89C-   20 79 F8    JSR   $F879
F89F-   D0 04       BNE   $F8A5
F8A1-   A0 FC       LDY   #$FC
F8A3-   A9 00       LDA   #$00
F8A5-   AA          TAX
F8A6-   BD A6 F9    LDA   $F9A6,X
F8A9-   85 2E       STA   $2E
F8AB-   29 03       AND   #$03
F8AD-   85 2F       STA   $2F
F8AF-   20 35 FC    JSR   $FC35
F8B2-   F0 18       BEQ   $F8CC
F8B4-   29 8F       AND   #$8F
F8B6-   AA          TAX
F8B7-   98          TYA
F8B8-   A0 03       LDY   #$03
F8BA-   E0 8A       CPX   #$8A
F8BC-   F0 0B       BEQ   $F8C9
F8BE-   4A          LSR
F8BF-   90 08       BCC   $F8C9
F8C1-   4A          LSR
F8C2-   4A          LSR
F8C3-   09 20       ORA   #$20
F8C5-   88          DEY
F8C6-   D0 FA       BNE   $F8C2
F8C8-   C8          INY
F8C9-   88          DEY
F8CA-   D0 F2       BNE   $F8BE
F8CC-   60          RTS
F8CD-   FF
```

```
F8CE-   FF
F8CF-   FF
F8D0-   20 82 F8    JSR   $F882
F8D3-   48          PHA
F8D4-   B1 3A       LDA   ($3A),Y
F8D6-   20 DA FD    JSR   $FDDA
F8D9-   A2 01       LDX   #$01
F8DB-   20 4A F9    JSR   $F94A
F8DE-   C4 2F       CPY   $2F
F8E0-   C8          INY
F8E1-   90 F1       BCC   $F8D4
F8E3-   A2 03       LDX   #$03
F8E5-   C0 04       CPY   #$04
F8E7-   90 F2       BCC   $F8DB
F8E9-   68          PLA
F8EA-   A8          TAY
F8EB-   B9 C0 F9    LDA   $F9C0,Y
F8EE-   85 2C       STA   $2C
F8F0-   B9 00 FA    LDA   $FA00,Y
F8F3-   85 2D       STA   $2D
F8F5-   A9 00       LDA   #$00
F8F7-   A0 05       LDY   #$05
F8F9-   06 2D       ASL   $2D
F8FB-   26 2C       ROL   $2C
F8FD-   2A          ROL
F8FE-   88          DEY
F8FF-   D0 F8       BNE   $F8F9
F901-   69 BF       ADC   #$BF
F903-   20 ED FD    JSR   $FDED
F906-   CA          DEX
F907-   D0 EC       BNE   $F8F5
F909-   20 48 F9    JSR   $F948
F90C-   A4 2F       LDY   $2F
F90E-   A2 06       LDX   #$06
F910-   E0 03       CPX   #$03
F912-   F0 1C       BEQ   $F930
F914-   06 2E       ASL   $2E
F916-   90 0E       BCC   $F926
F918-   BD B9 F9    LDA   $F9B9,X
F91B-   20 ED FD    JSR   $FDED
```

```
F91E-   BD B3 F9    LDA   $F9B3,X
F921-   F0 03       BEQ   $F926
F923-   20 ED FD    JSR   $FDED
F926-   CA          DEX
F927-   D0 E7       BNE   $F910
F929-   60          RTS
F92A-   88          DEY
F92B-   30 E7       BMI   $F914
F92D-   20 DA FD    JSR   $FDDA
F930-   A5 2E       LDA   $2E
F932-   C9 E8       CMP   #$E8
F934-   B1 3A       LDA   ($3A),Y
F936-   90 F2       BCC   $F92A
F938-   20 56 F9    JSR   $F956
F93B-   AA          TAX
F93C-   E8          INX
F93D-   D0 01       BNE   $F940
F93F-   C8          INY
F940-   98          TYA
F941-   20 DA FD    JSR   $FDDA
F944-   8A          TXA
F945-   4C DA FD    JMP   $FDDA
F948-   A2 03       LDX   #$03
F94A-   A9 A0       LDA   #$A0
F94C-   20 ED FD    JSR   $FDED
F94F-   CA          DEX
F950-   D0 F8       BNE   $F94A
F952-   60          RTS
F953-   38          SEC
F954-   A5 2F       LDA   $2F
F956-   A4 3B       LDY   $3B
F958-   AA          TAX
F959-   10 01       BPL   $F95C
F95B-   88          DEY
F95C-   65 3A       ADC   $3A
F95E-   90 01       BCC   $F961
F960-   C8          INY
F961-   60          RTS
F962-   0F
F963-   22
```

```
F964-   FF
F965-   33
F966-   CB
F967-   62
F968-   FF
F969-   73
F96A-   03
F96B-   22
F96C-   FF
F96D-   33
F96E-   CB
F96F-   66 FF       ROR   $FF
F971-   77
F972-   0F
F973-   20 FF 33    JSR   $33FF
F976-   CB
F977-   60          RTS
F978-   FF
F979-   70 0F       BVS   $F98A
F97B-   22
F97C-   FF
F97D-   39 CB 66    AND   $66CB,Y
F980-   FF
F981-   7D 0B 22    ADC   $220B,X
F984-   FF
F985-   33
F986-   CB
F987-   A6 FF       LDX   $FF
F989-   73
F98A-   11 22       ORA   ($22),Y
F98C-   FF
F98D-   33
F98E-   CB
F98F-   A6 FF       LDX   $FF
F991-   87
F992-   01 22       ORA   ($22,X)
F994-   FF
F995-   33
F996-   CB
F997-   60          RTS
```

```
F998-   FF
F999-   70 01       BVS   $F99C
F99B-   22
F99C-   FF
F99D-   33
F99E-   CB
F99F-   60          RTS
F9A0-   FF
F9A1-   70 24       BVS   $F9C7
F9A3-   31 65       AND   ($65),Y
F9A5-   78          SEI
F9A6-   00          BRK
F9A7-   21 81       AND   ($81,X)
F9A9-   82
F9AA-   59 4D 91    EOR   $914D,Y
F9AD-   92 86       STA   ($86)
F9AF-   4A          LSR
F9B0-   85 9D       STA   $9D
F9B2-   49 5A       EOR   #$5A
F9B4-   D9 00 D8    CMP   $D800,Y
F9B7-   A4 A4       LDY   $A4
F9B9-   00          BRK
F9BA-   AC A9 AC    LDY   $ACA9
F9BD-   A3
F9BE-   A8          TAY
F9BF-   A4 1C       LDY   $1C
F9C1-   8A          TXA
F9C2-   1C 23 5D    TRB   $5D23
F9C5-   8B
F9C6-   1B
F9C7-   A1 9D       LDA   ($9D,X)
F9C9-   8A          TXA
F9CA-   1D 23 9D    ORA   $9D23,X
F9CD-   8B
F9CE-   1D A1 1C    ORA   $1CA1,X
F9D1-   29 19       AND   #$19
F9D3-   AE 69 A8    LDX   $A869
F9D6-   19 23 24    ORA   $2423,Y
F9D9-   53
F9DA-   1B
```

```
F9DB-   23
F9DC-   24 53       BIT   $53
F9DE-   19 A1 AD    ORA   $ADA1,Y
F9E1-   1A          INC
F9E2-   5B
F9E3-   5B
F9E4-   A5 69       LDA   $69
F9E6-   24 24       BIT   $24
F9E8-   AE AE A8    LDX   $A8AE
F9EB-   AD 29 8A    LDA   $8A29
F9EE-   7C 8B 15    JMP   ($158B,X)
F9F1-   9C 6D 9C    STZ   $9C6D
F9F4-   A5 69       LDA   $69
F9F6-   29 53       AND   #$53
F9F8-   84 13       STY   $13
F9FA-   34 11       BIT   $11,X
F9FC-   A5 69       LDA   $69
F9FE-   23
F9FF-   A0 D8       LDY   #$D8
FA01-   62
FA02-   5A          PHY
FA03-   48          PHA
FA04-   26 62       ROL   $62
FA06-   94 88       STY   $88,X
FA08-   54
FA09-   44
FA0A-   C8          INY
FA0B-   54
FA0C-   68          PLA
FA0D-   44
FA0E-   E8          INX
FA0F-   94 C4       STY   $C4,X
FA11-   B4 08       LDY   $08,X
FA13-   84 74       STY   $74
FA15-   B4 28       LDY   $28,X
FA17-   6E 74 F4    ROR   $F474
FA1A-   CC 4A 72    CPY   $724A
FA1D-   F2 A4       SBC   ($A4)
FA1F-   8A          TXA
FA20-   06 AA       ASL   $AA
```

```
FA22-   A2 A2       LDX   #$A2
FA24-   74 74       STZ   $74,X
FA26-   74 72       STZ   $72,X
FA28-   44
FA29-   68          PLA
FA2A-   B2 32       LDA   ($32)
FA2C-   B2 72       LDA   ($72)
FA2E-   22
FA2F-   72 1A       ADC   ($1A)
FA31-   1A          INC
FA32-   26 26       ROL   $26
FA34-   72 72       ADC   ($72)
FA36-   88          DEY
FA37-   C8          INY
FA38-   C4 CA       CPY   $CA
FA3A-   26 48       ROL   $48
FA3C-   44
FA3D-   44
FA3E-   A2 C8       LDX   #$C8
FA40-   48          PHA
FA41-   68          PLA
FA42-   68          PLA
FA43-   4C 06 C8    JMP   $C806
FA46-   FF
FA47-   85 44       STA   $44
FA49-   7A          PLY
FA4A-   FA          PLX
FA4B-   68          PLA
FA4C-   28          PLP
FA4D-   20 4A FF    JSR   $FF4A
FA50-   68          PLA
FA51-   85 3A       STA   $3A
FA53-   68          PLA
FA54-   85 3B       STA   $3B
FA56-   6C F0 03    JMP   ($03F0)
FA59-   20 82 F8    JSR   $F882
FA5C-   20 DA FA    JSR   $FADA
FA5F-   4C 65 FF    JMP   $FF65
FA62-   D8          CLD
FA63-   20 84 FE    JSR   $FE84
```

```
FA66-   20 2F FB   JSR   $FB2F
FA69-   20 93 FE   JSR   $FE93
FA6C-   20 89 FE   JSR   $FE89
FA6F-   20 1C C4   JSR   $C41C
FA72-   20 04 CC   JSR   $CC04
FA75-   9C FF 04   STZ   $04FF
FA78-   AD 5F C0   LDA   $C05F
FA7B-   20 BD FA   JSR   $FABD
FA7E-   2C 10 C0   BIT   $C010
FA81-   D8         CLD
FA82-   20 3A FF   JSR   $FF3A
FA85-   AD F3 03   LDA   $03F3
FA88-   49 A5      EOR   #$A5
FA8A-   CD F4 03   CMP   $03F4
FA8D-   D0 17      BNE   $FAA6
FA8F-   AD F2 03   LDA   $03F2
FA92-   D0 3B      BNE   $FACF
FA94-   A9 E0      LDA   #$E0
FA96-   CD F3 03   CMP   $03F3
FA99-   D0 34      BNE   $FACF
FA9B-   A0 03      LDY   #$03
FA9D-   8C F2 03   STY   $03F2
FAA0-   4C 00 E0   JMP   $E000
FAA3-   20 3A FF   JSR   $FF3A
FAA6-   20 CA FC   JSR   $FCCA
FAA9-   A2 05      LDX   #$05
FAAB-   BD FC FA   LDA   $FAFC,X
FAAE-   9D EF 03   STA   $03EF,X
FAB1-   CA         DEX
FAB2-   D0 F7      BNE   $FAAB
FAB4-   A9 C6      LDA   #$C6
FAB6-   80 5A      BRA   $FB12
FAB8-   8A         TXA
FAB9-   8B
FABA-   A5 AC      LDA   $AC
FABC-   00         BRK
FABD-   A9 FF      LDA   #$FF
FABF-   8D FB 04   STA   $04FB
FAC2-   0E 62 C0   ASL   $C062
FAC5-   2C 61 C0   BIT   $C061
```

```
FAC8-   10 64       BPL   $FB2E
FACA-   90 D7       BCC   $FAA3
FACC-   4C 7C C7    JMP   $C77C
FACF-   6C F2 03    JMP   ($03F2)
FAD2-   C1 D8       CMP   ($D8,X)
FAD4-   D9 D0 D3    CMP   $D3D0,Y
FAD7-   20 8E FD    JSR   $FD8E
FADA-   A9 45       LDA   #$45
FADC-   85 40       STA   $40
FADE-   A9 00       LDA   #$00
FAE0-   85 41       STA   $41
FAE2-   A2 FB       LDX   #$FB
FAE4-   A9 A0       LDA   #$A0
FAE6-   20 ED FD    JSR   $FDED
FAE9-   BD D7 F9    LDA   $F9D7,X
FAEC-   20 ED FD    JSR   $FDED
FAEF-   A9 BD       LDA   #$BD
FAF1-   20 ED FD    JSR   $FDED
FAF4-   B5 4A       LDA   $4A,X
FAF6-   80 0A       BRA   $FB02
FAF8-   74 74       STZ   $74,X
FAFA-   76 C6       ROR   $C6,X
FAFC-   00          BRK
FAFD-   59 FA 00    EOR   $00FA,Y
FB00-   E0 45       CPX   #$45
FB02-   20 DA FD    JSR   $FDDA
FB05-   E8          INX
FB06-   30 DC       BMI   $FAE4
FB08-   60          RTS
FB09-   C1 F0       CMP   ($F0,X)
FB0B-   F0 EC       BEQ   $FAF9
FB0D-   E5 A0       SBC   $A0
FB0F-   DD DB C4    CMP   $C4DB,X
FB12-   86 00       STX   $00
FB14-   85 01       STA   $01
FB16-   20 60 FB    JSR   $FB60
FB19-   6C 00 00    JMP   ($0000)
FB1C-   FF
FB1D-   FF
FB1E-   4C DE C7    JMP   $C7DE
```

```
FB21-   A0 00       LDY   #$00
FB23-   EA          NOP
FB24-   EA          NOP
FB25-   BD 64 C0    LDA   $C064,X
FB28-   10 04       BPL   $FB2E
FB2A-   C8          INY
FB2B-   D0 F8       BNE   $FB25
FB2D-   88          DEY
FB2E-   60          RTS
FB2F-   A9 00       LDA   #$00
FB31-   85 48       STA   $48
FB33-   AD 56 C0    LDA   $C056
FB36-   AD 54 C0    LDA   $C054
FB39-   AD 51 C0    LDA   $C051
FB3C-   A9 00       LDA   #$00
FB3E-   F0 0B       BEQ   $FB4B
FB40-   AD 50 C0    LDA   $C050
FB43-   AD 53 C0    LDA   $C053
FB46-   20 36 F8    JSR   $F836
FB49-   A9 14       LDA   #$14
FB4B-   85 22       STA   $22
FB4D-   EA          NOP
FB4E-   EA          NOP
FB4F-   20 0A CE    JSR   $CE0A
FB52-   80 05       BRA   $FB59
FB54-   09 80       ORA   #$80
FB56-   4C 54 CD    JMP   $CD54
FB59-   A9 17       LDA   #$17
FB5B-   85 25       STA   $25
FB5D-   4C 22 FC    JMP   $FC22
FB60-   20 58 FC    JSR   $FC58
FB63-   A0 09       LDY   #$09
FB65-   B9 02 FD    LDA   $FD02,Y
FB68-   99 0D 04    STA   $040D,Y
FB6B-   88          DEY
FB6C-   D0 F7       BNE   $FB65
FB6E-   60          RTS
FB6F-   AD F3 03    LDA   $03F3
FB72-   49 A5       EOR   #$A5
FB74-   8D F4 03    STA   $03F4
```

```
FB77-   60          RTS
FB78-   C9 8D       CMP   #$8D
FB7A-   D0 18       BNE   $FB94
FB7C-   AC 00 C0    LDY   $C000
FB7F-   10 13       BPL   $FB94
FB81-   C0 93       CPY   #$93
FB83-   D0 0F       BNE   $FB94
FB85-   2C 10 C0    BIT   $C010
FB88-   AC 00 C0    LDY   $C000
FB8B-   10 FB       BPL   $FB88
FB8D-   C0 83       CPY   #$83
FB8F-   F0 03       BEQ   $FB94
FB91-   2C 10 C0    BIT   $C010
FB94-   2C 7B 06    BIT   $067B
FB97-   30 64       BMI   $FBFD
FB99-   89 60       BIT   #$60
FB9B-   F0 B7       BEQ   $FB54
FB9D-   20 B8 C3    JSR   $C3B8
FBA0-   EE 7B 05    INC   $057B
FBA3-   AD 7B 05    LDA   $057B
FBA6-   2C 1F C0    BIT   $C01F
FBA9-   30 05       BMI   $FBB0
FBAB-   8D 7B 04    STA   $047B
FBAE-   85 24       STA   $24
FBB0-   80 46       BRA   $FBF8
FBB2-   FF
FBB3-   06 10       ASL   $10
FBB5-   06 C9       ASL   $C9
FBB7-   A0 90       LDY   #$90
FBB9-   02
FBBA-   25 32       AND   $32
FBBC-   4C F6 FD    JMP   $FDF6
FBBF-   FF
FBC0-   00          BRK
FBC1-   48          PHA
FBC2-   4A          LSR
FBC3-   29 03       AND   #$03
FBC5-   09 04       ORA   #$04
FBC7-   85 29       STA   $29
FBC9-   68          PLA
```

```
FBCA-   29 18       AND   #$18
FBCC-   90 02       BCC   $FBD0
FBCE-   69 7F       ADC   #$7F
FBD0-   85 28       STA   $28
FBD2-   0A          ASL
FBD3-   0A          ASL
FBD4-   05 28       ORA   $28
FBD6-   85 28       STA   $28
FBD8-   60          RTS
FBD9-   C9 87       CMP   #$87
FBDB-   D0 12       BNE   $FBEF
FBDD-   A9 40       LDA   #$40
FBDF-   20 A8 FC    JSR   $FCA8
FBE2-   A0 C0       LDY   #$C0
FBE4-   A9 0C       LDA   #$0C
FBE6-   20 A8 FC    JSR   $FCA8
FBE9-   AD 30 C0    LDA   $C030
FBEC-   88          DEY
FBED-   D0 F5       BNE   $FBE4
FBEF-   60          RTS
FBF0-   A4 24       LDY   $24
FBF2-   91 28       STA   ($28),Y
FBF4-   E6 24       INC   $24
FBF6-   A5 24       LDA   $24
FBF8-   C5 21       CMP   $21
FBFA-   B0 66       BCS   $FC62
FBFC-   60          RTS
FBFD-   C9 A0       CMP   #$A0
FBFF-   B0 EF       BCS   $FBF0
FC01-   A8          TAY
FC02-   10 EC       BPL   $FBF0
FC04-   C9 8D       CMP   #$8D
FC06-   F0 6B       BEQ   $FC73
FC08-   C9 8A       CMP   #$8A
FC0A-   F0 5A       BEQ   $FC66
FC0C-   C9 88       CMP   #$88
FC0E-   D0 C9       BNE   $FBD9
FC10-   20 E2 FE    JSR   $FEE2
FC13-   10 E7       BPL   $FBFC
FC15-   A5 21       LDA   $21
```

```
FC17-   20 EB FE    JSR   $FEEB
FC1A-   A5 22       LDA   $22
FC1C-   C5 25       CMP   $25
FC1E-   B0 DC       BCS   $FBFC
FC20-   C6 25       DEC   $25
FC22-   80 62       BRA   $FC86
FC24-   20 C1 FB    JSR   $FBC1
FC27-   A5 20       LDA   $20
FC29-   2C 1F C0    BIT   $C01F
FC2C-   10 02       BPL   $FC30
FC2E-   4A          LSR
FC2F-   18          CLC
FC30-   65 28       ADC   $28
FC32-   85 28       STA   $28
FC34-   60          RTS
FC35-   98          TYA
FC36-   A2 16       LDX   #$16
FC38-   DD FE FE    CMP   $FEFE,X
FC3B-   F0 43       BEQ   $FC80
FC3D-   CA          DEX
FC3E-   10 F8       BPL   $FC38
FC40-   60          RTS
FC41-   FF
FC42-   80 19       BRA   $FC5D
FC44-   A5 25       LDA   $25
FC46-   48          PHA
FC47-   20 24 FC    JSR   $FC24
FC4A-   20 9E FC    JSR   $FC9E
FC4D-   A0 00       LDY   #$00
FC4F-   68          PLA
FC50-   1A          INC
FC51-   C5 23       CMP   $23
FC53-   90 F1       BCC   $FC46
FC55-   B0 CB       BCS   $FC22
FC57-   FF
FC58-   20 A5 CD    JSR   $CDA5
FC5B-   80 E7       BRA   $FC44
FC5D-   20 9D CC    JSR   $CC9D
FC60-   80 E2       BRA   $FC44
FC62-   80 0F       BRA   $FC73
```

```
FC64-   FA          PLX
FC65-   FA          PLX
FC66-   E6 25       INC   $25
FC68-   A5 25       LDA   $25
FC6A-   C5 23       CMP   $23
FC6C-   90 1A       BCC   $FC88
FC6E-   C6 25       DEC   $25
FC70-   4C 35 CB    JMP   $CB35
FC73-   20 E9 FE    JSR   $FEE9
FC76-   2C FB 04    BIT   $04FB
FC79-   10 0A       BPL   $FC85
FC7B-   20 44 FD    JSR   $FD44
FC7E-   80 E6       BRA   $FC66
FC80-   BD 15 FF    LDA   $FF15,X
FC83-   A0 00       LDY   #$00
FC85-   60          RTS
FC86-   A5 25       LDA   $25
FC88-   8D FB 05    STA   $05FB
FC8B-   80 97       BRA   $FC24
FC8D-   20 9D CC    JSR   $CC9D
FC90-   A9 A0       LDA   #$A0
FC92-   2C 7B 06    BIT   $067B
FC95-   30 02       BMI   $FC99
FC97-   25 32       AND   $32
FC99-   4C C2 CB    JMP   $CBC2
FC9C-   80 EF       BRA   $FC8D
FC9E-   80 F0       BRA   $FC90
FCA0-   A0 00       LDY   #$00
FCA2-   80 EC       BRA   $FC90
FCA4-   7C 2A CD    JMP   ($CD2A,X)
FCA7-   FF
FCA8-   38          SEC
FCA9-   48          PHA
FCAA-   E9 01       SBC   #$01
FCAC-   D0 FC       BNE   $FCAA
FCAE-   68          PLA
FCAF-   E9 01       SBC   #$01
FCB1-   D0 F6       BNE   $FCA9
FCB3-   60          RTS
FCB4-   E6 42       INC   $42
```

```
FCB6-   D0 02       BNE   $FCBA
FCB8-   E6 43       INC   $43
FCBA-   A5 3C       LDA   $3C
FCBC-   C5 3E       CMP   $3E
FCBE-   A5 3D       LDA   $3D
FCC0-   E5 3F       SBC   $3F
FCC2-   E6 3C       INC   $3C
FCC4-   D0 02       BNE   $FCC8
FCC6-   E6 3D       INC   $3D
FCC8-   60          RTS
FCC9-   60          RTS
FCCA-   A0 B0       LDY   #$B0
FCCC-   64 3C       STZ   $3C
FCCE-   A2 BF       LDX   #$BF
FCD0-   86 3D       STX   $3D
FCD2-   A9 A0       LDA   #$A0
FCD4-   91 3C       STA   ($3C),Y
FCD6-   88          DEY
FCD7-   91 3C       STA   ($3C),Y
FCD9-   CA          DEX
FCDA-   E0 01       CPX   #$01
FCDC-   D0 F2       BNE   $FCD0
FCDE-   8D 01 C0    STA   $C001
FCE1-   AD 55 C0    LDA   $C055
FCE4-   38          SEC
FCE5-   A2 88       LDX   #$88
FCE7-   BD 27 CB    LDA   $CB27,X
FCEA-   90 0A       BCC   $FCF6
FCEC-   DD 77 04    CMP   $0477,X
FCEF-   18          CLC
FCF0-   D0 04       BNE   $FCF6
FCF2-   E0 82       CPX   #$82
FCF4-   90 06       BCC   $FCFC
FCF6-   9D 77 04    STA   $0477,X
FCF9-   CA          DEX
FCFA-   D0 EB       BNE   $FCE7
FCFC-   AD 54 C0    LDA   $C054
FCFF-   8D 00 C0    STA   $C000
FD02-   60          RTS
FD03-   C1 F0       CMP   ($F0,X)
```

```
FD05-   F0 EC       BEQ   $FCF3
FD07-   E5 A0       SBC   $A0
FD09-   AF
FD0A-   AF
FD0B-   E3
FD0C-   A4 24       LDY   $24
FD0E-   B1 28       LDA   ($28),Y
FD10-   EA          NOP
FD11-   EA          NOP
FD12-   EA          NOP
FD13-   EA          NOP
FD14-   EA          NOP
FD15-   EA          NOP
FD16-   EA          NOP
FD17-   EA          NOP
FD18-   6C 38 00    JMP   ($0038)
FD1B-   91 28       STA   ($28),Y
FD1D-   20 4C CC    JSR   $CC4C
FD20-   20 70 CC    JSR   $CC70
FD23-   10 FB       BPL   $FD20
FD25-   48          PHA
FD26-   A9 08       LDA   #$08
FD28-   2C FB 04    BIT   $04FB
FD2B-   D0 1D       BNE   $FD4A
FD2D-   68          PLA
FD2E-   C9 9B       CMP   #$9B
FD30-   D0 06       BNE   $FD38
FD32-   4C CC CC    JMP   $CCCC
FD35-   4C ED CC    JMP   $CCED
FD38-   2C 7B 06    BIT   $067B
FD3B-   30 07       BMI   $FD44
FD3D-   C9 95       CMP   #$95
FD3F-   D0 03       BNE   $FD44
FD41-   20 1D CC    JSR   $CC1D
FD44-   48          PHA
FD45-   A9 08       LDA   #$08
FD47-   0C FB 04    TSB   $04FB
FD4A-   68          PLA
FD4B-   60          RTS
FD4C-   EA          NOP
```

```
FD4D-   20 A6 C3    JSR   $C3A6
FD50-   C9 88       CMP   #$88
FD52-   F0 1D       BEQ   $FD71
FD54-   C9 98       CMP   #$98
FD56-   F0 0A       BEQ   $FD62
FD58-   E0 F8       CPX   #$F8
FD5A-   90 03       BCC   $FD5F
FD5C-   20 3A FF    JSR   $FF3A
FD5F-   E8          INX
FD60-   D0 13       BNE   $FD75
FD62-   A9 DC       LDA   #$DC
FD64-   20 A6 C3    JSR   $C3A6
FD67-   20 8E FD    JSR   $FD8E
FD6A-   A5 33       LDA   $33
FD6C-   20 ED FD    JSR   $FDED
FD6F-   A2 01       LDX   #$01
FD71-   8A          TXA
FD72-   F0 F3       BEQ   $FD67
FD74-   CA          DEX
FD75-   20 ED CC    JSR   $CCED
FD78-   C9 95       CMP   #$95
FD7A-   D0 08       BNE   $FD84
FD7C-   20 1D CC    JSR   $CC1D
FD7F-   EA          NOP
FD80-   EA          NOP
FD81-   EA          NOP
FD82-   EA          NOP
FD83-   EA          NOP
FD84-   9D 00 02    STA   $0200,X
FD87-   C9 8D       CMP   #$8D
FD89-   D0 C2       BNE   $FD4D
FD8B-   20 9C FC    JSR   $FC9C
FD8E-   A9 8D       LDA   #$8D
FD90-   D0 5B       BNE   $FDED
FD92-   A4 3D       LDY   $3D
FD94-   A6 3C       LDX   $3C
FD96-   20 8E FD    JSR   $FD8E
FD99-   20 40 F9    JSR   $F940
FD9C-   A0 00       LDY   #$00
FD9E-   A9 AD       LDA   #$AD
```

```
FDA0-   4C ED FD    JMP   $FDED
FDA3-   A5 3C       LDA   $3C
FDA5-   09 07       ORA   #$07
FDA7-   85 3E       STA   $3E
FDA9-   A5 3D       LDA   $3D
FDAB-   85 3F       STA   $3F
FDAD-   A5 3C       LDA   $3C
FDAF-   29 07       AND   #$07
FDB1-   D0 03       BNE   $FDB6
FDB3-   20 92 FD    JSR   $FD92
FDB6-   A9 A0       LDA   #$A0
FDB8-   20 ED FD    JSR   $FDED
FDBB-   B1 3C       LDA   ($3C),Y
FDBD-   20 DA FD    JSR   $FDDA
FDC0-   20 BA FC    JSR   $FCBA
FDC3-   90 E8       BCC   $FDAD
FDC5-   60          RTS
FDC6-   4A          LSR
FDC7-   90 EA       BCC   $FDB3
FDC9-   4A          LSR
FDCA-   4A          LSR
FDCB-   A5 3E       LDA   $3E
FDCD-   90 02       BCC   $FDD1
FDCF-   49 FF       EOR   #$FF
FDD1-   65 3C       ADC   $3C
FDD3-   48          PHA
FDD4-   A9 BD       LDA   #$BD
FDD6-   20 ED FD    JSR   $FDED
FDD9-   68          PLA
FDDA-   48          PHA
FDDB-   4A          LSR
FDDC-   4A          LSR
FDDD-   4A          LSR
FDDE-   4A          LSR
FDDF-   20 E5 FD    JSR   $FDE5
FDE2-   68          PLA
FDE3-   29 0F       AND   #$0F
FDE5-   09 B0       ORA   #$B0
FDE7-   C9 BA       CMP   #$BA
FDE9-   90 02       BCC   $FDED
```

```
FDEB-   69 06       ADC   #$06
FDED-   6C 36 00    JMP   ($0036)
FDF0-   2C 7B 06    BIT   $067B
FDF3-   4C B4 FB    JMP   $FBB4
FDF6-   84 35       STY   $35
FDF8-   48          PHA
FDF9-   20 78 FB    JSR   $FB78
FDFC-   68          PLA
FDFD-   A4 35       LDY   $35
FDFF-   60          RTS
FE00-   C6 34       DEC   $34
FE02-   F0 9F       BEQ   $FDA3
FE04-   CA          DEX
FE05-   D0 16       BNE   $FE1D
FE07-   C9 BA       CMP   #$BA
FE09-   D0 BB       BNE   $FDC6
FE0B-   85 31       STA   $31
FE0D-   A5 3E       LDA   $3E
FE0F-   91 40       STA   ($40),Y
FE11-   E6 40       INC   $40
FE13-   D0 02       BNE   $FE17
FE15-   E6 41       INC   $41
FE17-   60          RTS
FE18-   A4 34       LDY   $34
FE1A-   B9 FF 01    LDA   $01FF,Y
FE1D-   85 31       STA   $31
FE1F-   60          RTS
FE20-   A2 01       LDX   #$01
FE22-   B5 3E       LDA   $3E,X
FE24-   95 42       STA   $42,X
FE26-   95 44       STA   $44,X
FE28-   CA          DEX
FE29-   10 F7       BPL   $FE22
FE2B-   60          RTS
FE2C-   B1 3C       LDA   ($3C),Y
FE2E-   91 42       STA   ($42),Y
FE30-   20 B4 FC    JSR   $FCB4
FE33-   90 F7       BCC   $FE2C
FE35-   60          RTS
FE36-   B1 3C       LDA   ($3C),Y
```

```
FE38-   D1 42       CMP   ($42),Y
FE3A-   F0 1C       BEQ   $FE58
FE3C-   20 92 FD    JSR   $FD92
FE3F-   B1 3C       LDA   ($3C),Y
FE41-   20 DA FD    JSR   $FDDA
FE44-   A9 A0       LDA   #$A0
FE46-   20 ED FD    JSR   $FDED
FE49-   A9 A8       LDA   #$A8
FE4B-   20 ED FD    JSR   $FDED
FE4E-   B1 42       LDA   ($42),Y
FE50-   20 DA FD    JSR   $FDDA
FE53-   A9 A9       LDA   #$A9
FE55-   20 ED FD    JSR   $FDED
FE58-   20 B4 FC    JSR   $FCB4
FE5B-   90 D9       BCC   $FE36
FE5D-   60          RTS
FE5E-   20 75 FE    JSR   $FE75
FE61-   A9 14       LDA   #$14
FE63-   48          PHA
FE64-   20 D0 F8    JSR   $F8D0
FE67-   20 53 F9    JSR   $F953
FE6A-   85 3A       STA   $3A
FE6C-   84 3B       STY   $3B
FE6E-   68          PLA
FE6F-   38          SEC
FE70-   E9 01       SBC   #$01
FE72-   D0 EF       BNE   $FE63
FE74-   60          RTS
FE75-   8A          TXA
FE76-   F0 07       BEQ   $FE7F
FE78-   B5 3C       LDA   $3C,X
FE7A-   95 3A       STA   $3A,X
FE7C-   CA          DEX
FE7D-   10 F9       BPL   $FE78
FE7F-   60          RTS
FE80-   A0 3F       LDY   #$3F
FE82-   D0 02       BNE   $FE86
FE84-   A0 FF       LDY   #$FF
FE86-   84 32       STY   $32
FE88-   60          RTS
```

```
FE89-   A9 00       LDA   #$00
FE8B-   85 3E       STA   $3E
FE8D-   A2 38       LDX   #$38
FE8F-   A0 1B       LDY   #$1B
FE91-   D0 08       BNE   $FE9B
FE93-   A9 00       LDA   #$00
FE95-   85 3E       STA   $3E
FE97-   A2 36       LDX   #$36
FE99-   A0 F0       LDY   #$F0
FE9B-   A5 3E       LDA   $3E
FE9D-   29 0F       AND   #$0F
FE9F-   D0 06       BNE   $FEA7
FEA1-   C0 1B       CPY   #$1B
FEA3-   F0 39       BEQ   $FEDE
FEA5-   80 1B       BRA   $FEC2
FEA7-   09 C0       ORA   #$C0
FEA9-   A0 00       LDY   #$00
FEAB-   94 00       STY   $00,X
FEAD-   95 01       STA   $01,X
FEAF-   60          RTS
FEB0-   4C 00 E0    JMP   $E000
FEB3-   4C 03 E0    JMP   $E003
FEB6-   20 75 FE    JSR   $FE75
FEB9-   20 3F FF    JSR   $FF3F
FEBC-   6C 3A 00    JMP   ($003A)
FEBF-   4C D7 FA    JMP   $FAD7
FEC2-   3A          DEC
FEC3-   8D FB 07    STA   $07FB
FEC6-   A9 F7       LDA   #$F7
FEC8-   80 04       BRA   $FECE
FECA-   4C F8 03    JMP   $03F8
FECD-   60          RTS
FECE-   8D 7B 06    STA   $067B
FED1-   8D 0E C0    STA   $C00E
FED4-   0C FB 04    TSB   $04FB
FED7-   DA          PHX
FED8-   5A          PHY
FED9-   20 CD CD    JSR   $CDCD
FEDC-   7A          PLY
FEDD-   FA          PLX
```

```
FEDE-   A9 FD       LDA   #$FD
FEE0-   80 C9       BRA   $FEAB
FEE2-   5A          PHY
FEE3-   20 9D CC    JSR   $CC9D
FEE6-   88          DEY
FEE7-   80 05       BRA   $FEEE
FEE9-   A9 01       LDA   #$01
FEEB-   3A          DEC
FEEC-   5A          PHY
FEED-   A8          TAY
FEEE-   20 AD CC    JSR   $CCAD
FEF1-   7A          PLY
FEF2-   AD 7B 05    LDA   $057B
FEF5-   60          RTS
FEF6-   20 00 FE    JSR   $FE00
FEF9-   68          PLA
FEFA-   68          PLA
FEFB-   D0 6C       BNE   $FF69
FEFD-   60          RTS
FEFE-   12 14       ORA   ($14)
FF00-   1A          INC
FF01-   1C 32 34    TRB   $3432
FF04-   3A          DEC
FF05-   3C 52 5A    BIT   $5A52,X
FF08-   64 72       STZ   $72
FF0A-   74 7A       STZ   $7A,X
FF0C-   7C 89 92    JMP   ($9289,X)
FF0F-   9C 9E B2    STZ   $B29E
FF12-   D2 F2       CMP   ($F2)
FF14-   FC
FF15-   38          SEC
FF16-   FB
FF17-   37
FF18-   FB
FF19-   39 21 36    AND   $3621,Y
FF1C-   21 3A       AND   ($3A,X)
FF1E-   F8          SED
FF1F-   FA          PLX
FF20-   3B
FF21-   FA          PLX
```

```
FF22-   F9 22 21    SBC   $2122,Y
FF25-   3C FA FA    BIT   $FAFA,X
FF28-   3D 3E 3F    AND   $3F3E,X
FF2B-   FC
FF2C-   00          BRK
FF2D-   A9 C5       LDA   #$C5
FF2F-   20 ED FD    JSR   $FDED
FF32-   A9 D2       LDA   #$D2
FF34-   20 ED FD    JSR   $FDED
FF37-   20 ED FD    JSR   $FDED
FF3A-   A9 87       LDA   #$87
FF3C-   4C ED FD    JMP   $FDED
FF3F-   A5 48       LDA   $48
FF41-   48          PHA
FF42-   A5 45       LDA   $45
FF44-   A6 46       LDX   $46
FF46-   A4 47       LDY   $47
FF48-   28          PLP
FF49-   60          RTS
FF4A-   85 45       STA   $45
FF4C-   86 46       STX   $46
FF4E-   84 47       STY   $47
FF50-   08          PHP
FF51-   68          PLA
FF52-   85 48       STA   $48
FF54-   BA          TSX
FF55-   86 49       STX   $49
FF57-   D8          CLD
FF58-   60          RTS
FF59-   20 84 FE    JSR   $FE84
FF5C-   20 2F FB    JSR   $FB2F
FF5F-   20 93 FE    JSR   $FE93
FF62-   20 89 FE    JSR   $FE89
FF65-   D8          CLD
FF66-   20 3A FF    JSR   $FF3A
FF69-   A9 AA       LDA   #$AA
FF6B-   85 33       STA   $33
FF6D-   20 67 FD    JSR   $FD67
FF70-   20 C7 FF    JSR   $FFC7
FF73-   20 A7 FF    JSR   $FFA7
```

```
FF76-   84 34       STY   $34
FF78-   A0 13       LDY   #$13
FF7A-   88          DEY
FF7B-   30 E8       BMI   $FF65
FF7D-   D9 CD FF    CMP   $FFCD,Y
FF80-   D0 F8       BNE   $FF7A
FF82-   20 BE FF    JSR   $FFBE
FF85-   A4 34       LDY   $34
FF87-   4C 73 FF    JMP   $FF73
FF8A-   A2 03       LDX   #$03
FF8C-   0A          ASL
FF8D-   0A          ASL
FF8E-   0A          ASL
FF8F-   0A          ASL
FF90-   0A          ASL
FF91-   26 3E       ROL   $3E
FF93-   26 3F       ROL   $3F
FF95-   CA          DEX
FF96-   10 F8       BPL   $FF90
FF98-   A5 31       LDA   $31
FF9A-   D0 06       BNE   $FFA2
FF9C-   B5 3F       LDA   $3F,X
FF9E-   95 3D       STA   $3D,X
FFA0-   95 41       STA   $41,X
FFA2-   E8          INX
FFA3-   F0 F3       BEQ   $FF98
FFA5-   D0 06       BNE   $FFAD
FFA7-   A2 00       LDX   #$00
FFA9-   86 3E       STX   $3E
FFAB-   86 3F       STX   $3F
FFAD-   B9 00 02    LDA   $0200,Y
FFB0-   C8          INY
FFB1-   20 99 C3    JSR   $C399
FFB4-   49 B0       EOR   #$B0
FFB6-   C9 0A       CMP   #$0A
FFB8-   90 D0       BCC   $FF8A
FFBA-   80 37       BRA   $FFF3
FFBC-   FF
FFBD-   41 A9       EOR   ($A9,X)
FFBF-   FE 48 B9    INC   $B948,X
```

```
FFC2-   E0 FF       CPX   #$FF
FFC4-   48          PHA
FFC5-   A5 31       LDA   $31
FFC7-   A0 00       LDY   #$00
FFC9-   84 31       STY   $31
FFCB-   60          RTS
FFCC-   EA          NOP
FFCD-   BC B2 BE    LDY   $BEB2,X
FFD0-   EF
FFD1-   C4 A9       CPY   $A9
FFD3-   BB
FFD4-   A6 A4       LDX   $A4
FFD6-   06 95       ASL   $95
FFD8-   07
FFD9-   02
FFDA-   05 00       ORA   $00
FFDC-   93
FFDD-   A7
FFDE-   C6 99       DEC   $99
FFE0-   B2 C9       LDA   ($C9)
FFE2-   BE 35 8C    LDX   $8C35,Y
FFE5-   96 AF       STX   $AF,Y
FFE7-   17
FFE8-   17
FFE9-   2B
FFEA-   1F
FFEB-   83
FFEC-   7F
FFED-   5D B5 17    EOR   $17B5,X
FFF0-   17
FFF1-   F5 03       SBC   $03,X
FFF3-   69 88       ADC   #$88
FFF5-   C9 FA       CMP   #$FA
FFF7-   B0 91       BCS   $FF8A
FFF9-   60          RTS
FFFA-   FB
FFFB-   03
FFFC-   62
FFFD-   FA          PLX
FFFE-   03
FFFF-   C8          INY
```

## Stichwortverzeichnis


*****

*** >32767 ERR 91

**3**

3D-Darstellungen 212

**4**

40 Zeichendarstellung 65

**8**

80 Zeichendarstellung 65

**A**

Addition 88
Akustikkoppler 32
AND 114
APPEND 228
Apple III 19, 68
Applesoft-BASIC 71, 80
Architekturplanung 179
Argument 148
ASCII-Code der deutschen Tastenbelegung 277
ASCII-Code der US-Tastenbelegung 279
Ausdrucken 51
AUTO 96

**B**

Backup 239
BASIC 37, 71
BASIC-Betriebsarten 84
BASIC-Compiler 84
BASIC-Interpreter 84
BASIC-Startdiskette unter DOS 3.3 78
BASIC-Startdiskette unter ProDOS 73
BASIC-Versionen 71
Betriebsart
 -, Direktanweisung 85
 -, Programmieren 91
Bilder-Animation 210
Bildschirmformat 65
Bildschirmtext 56
Boole'sche Algebra 114

Stichwortverzeichnis

**C**
CAT 244
CATALOG 221, 240, 244
Centronic- Schnittstelle 51
CHR$ 226
CLEAR 119
CLOSE 226
CLR 119
CONT 142
CONTROL-Taste 27
CPU 42, 259
Cursor 64

**D**
Darstellungsvarianten 135
DATA und READ 118
Dateien
-, kopieren 237
-, löschen 238
-, umbenennen 238
Datenfernübertragung 53
Datenverwaltung 38
DEL 101
DELETE 223
Dezimalzahlen 89
Dialogprogrammierung 137
Direktanweisung 84
Diskette 217
Disketten
-, fehler 243
-, format 240
-, name 240
-, überprüfen 243
Diskettenlaufwerk
extern 54
Division 88
DOS 72, 217
DOS 3.2 243
DOS 3.3 220

**E**
Editieren des Programms 98
Ein- und Ausgabefunktionen 129
Eingabeformat in ProDOS 250

Eingabefunktion
 -, GET 137
 -, INPUT 138
Exponentenschreibweise 104
Exponenten 89

F
Fehlerbehandlung 143
Fehlermeldung 85
Fehlermeldungen in Applesoft-BASIC 270
Fehlermeldungen in Integer-BASIC 272
Festplattenlaufwerk 55, 235
Festwertspeicher 44
File 242
Filepuffer bei der Datenübertragung 254
Filetyp 244
Flachbildschirm 50
FLASH 136
Floppydisk 217, 220
Formatieren
 -, einer Diskette 240
FP und INT in ProDOS 250
Funktionen 148

G
Ganzzahl-BASIC 79
Garantieanspruch 41
GOSUB 124
GOTO 120
Grafik mit Applesoft-BASIC 197
Grafiksymbole 178

H
HELLO 221
Hochauflösende Grafik 204
HOME 91, 102
HTAB 133

I
IF...THEN-Anweisung 128
IN# 147
Inhaltsverzeichnis
 -, einer Diskette 240

INPUT 107
Integer-BASIC 71, 79

**J**
Joystick 31
Joystick und Paddleansteuerung 146

**K**
Kompatibilität zu anderen Apple-Computern 68
Kopfhöreranschluß 28
Kopieren
 -, einer Diskette 239
 -, von Dateien 237

**L**
Laden und Abspeichern von Daten 248
Laden von ProDOS 236
LET 111
LIST 92, 95
LOAD 224
LOCK 224, 239
LOGO 37
Löschen
 -, einzelner Zeichen 100
 -, ganzer Zeilen 100

**M**
Macintosh 20
Maus 171
Menü 236
Mikroprozessor 43
Modem 32
MON 225, 227
Monitore 47
MousePaint 175
Multiplikation 88

**N**
NEW 92, 94
Niedrigauflösende Grafik 197
NOMON 225
NORMAL 136
NOT 114

**O**

ON .. GOTO 121
ON GOSUB 127
ONERR GOTO 144
OPEN 226
Operatoren 111
OR 114

**P**

PAL-Modulator/Adapter 35
Peripheriegeräte 50
Pfadnamen 242
POP 126
POSITION 228
Potenzieren 88
PR# 147
PR# und IN# in ProDOS 254
Prefix 241
PRINT 86
ProDOS
 -, Startdiskette für BASIC 72
Programmaufbau und Ablauf 120
Programmieren in BASIC 80
Programmschleifen 123
Programmsteuerungen 120
Programmzeilennummer 94

**R**

RAM 44
REM 97
Reservierte Worte 111
Reset-Taste 27
RESTORE 119
RESUME 144
RETURN 94
ROM 44

**S**

SAVE 223
Schreib-/Lesekopf 44
Schreibschutz
 -, setzen, aufheben 239
Selektion von Peripheriegeräten 147
SOS 68

SPEED 142
Speicheradressen anzeigen 262
Speicherinhalte verändern 263
Speichern von Grafikseiten 209
Spiele 36
Sprachgenerator 56
Sprünganweisung 120
Startdiskette 72
Starten des System-Monitors 262
Starten von Monitorprogrammen 266
STARTUP 247
Startup-Programm 247
Steuerknüppel 54
STOP 141
Subdirectory 242
Subtraktion 88
Suffix 247
SYNTAX ERROR 85
System-Dienstprogramme 236, 237
Systemmeldung 63

**T**
TAB 133
Tabellen 114
Tabellenkalkulation 38
Tastatur 26, 46, 66
Text-Funktionen 155
Textverarbeitung 38
THEN END 93

**U**
Umschalten von Applesoft- auf Integer-BASIC 82
UNLOCK 224
Untermenü 237
Unterprogramm 236
US-Zeichensatz 66

**V**
VERIFY 224
Verschieben eines Speicherbereichs 265
Videosignalbuchse 33
Volumen 221

**W**
WAIT 143
Warteschleifen 141
Wurzelfunktion 148

**Z**
Zahlen
-, Texte und Variable 103
Zeilennumerierung 95

# Weitere Fachbücher aus unserem Verlagsprogramm

**Personal Computer Lexikon**
**1982, 136 Seiten**
Über 1000 Suchbegriffe aus Hard- und Software · deutsch/englisch · ausführlicher Artikel zu jedem Suchbegriff · englisch/deutsch Register im Anhang · der ideale Einstieg ins Homecomputing · das unentbehrliche Nachschlagewerk für den Profi.
**Best.-Nr. MT 390, DM 19,80 (Sfr. 18,50/öS 154,40)**

J. Willis/M. Miller
**Computertechnik ohne Geheimnisse**
**November 1984, 313 Seiten**
Eine allgemeine Einführung in die Welt der Computertechnik · die wichtigsten Grundbegriffe knapp und übersichtlich dargestellt · nützliche Informationen für die richtige Kaufentscheidung · mit einer aktuellen Marktübersicht der gängisten Rechnermodelle und deren Zubehör.
**Best.-Nr. MT 716, DM 42,— (Sfr. 38,60/öS 327,60)**

W. Pest: **Computerchinesisch für Einsteiger**
**Juli 1984, 107 Seiten**
Ein praxisnahes Lexikon, das Personal Computer-Benutzern und solchen, die es werden wollen, das Lesen von Fachzeitschriften, Büchern, Bedienungsanleitungen und Datenblättern erleichtert · über 1000 häufig benötigte Fachbegriffe klar und verständlich erläutert · mit zahlreichen Abbildungen.
**Best.-Nr. MT 690, DM 28,— (Sfr. 25,90/öS 218,40)**

Dr. M. Henk: **Die IBM-Personal Computer**
**2. überarbeitete Auflage Februar 1985, 350 Seiten**
Die IBM-PC's in ihrer Hard- und Software · Betriebssysteme · Programmiersprachen · Textverarbeitung · Tabellen- und Planungsprogramme · zusätzliche Hardware- und Softwareprodukte · IBM-PC-kompatible Rechner und Mitbewerbersysteme.
**Best.-Nr. MT 630, DM 58,— (Sfr. 53,40/öS 452,40)**

J. E. Kelley jr.: **Die Welt des IBM-PC**
**Juli 1984, 444 Seiten**
Ein praktisches Handbuch für den mühelosen Umgang mit Ihrem IBM-PC · ausführlichst erläuterte Befehle · Programmentwicklung in Basic · Menüs und andere Hilfsmittel · Grafikprozeduren · mit Beispielen auf Diskette (im Buch enthalten!) ·
**Best.-Nr. MT 636, DM 88,— (Sfr. 81,—/öS 686,40)**

R. Traister: **Spielen mit dem IBM-PC**
**1984, 318 Seiten**
33 ausgewählte Spieleprogramme · Würfelspiele, Russisches Roulette für den risikofreudigen Spieler · Intelligenztest, Biorhythmen und chinesische Astrologie zur persönlichen Beratung · der Schachteufel · das Morsealphabeth · Wortspiele · ein Sammelsurium an Tüfteleien und Action-Spiele für ein, zwei oder mehrere Spieler.
**Best.-Nr. MT 661, DM 58,— (Sfr. 53,40/öS 452,40)**
**Best.-Nr. MT 684 (Beispiele auf Diskette)**
**DM 48,— (Sfr. 48,—/öS 374,40)**

R. J. Traister: **Grafik-Programme für den IBM-PC**
**Juli 1984, 271 Seiten**
Alles über das grafische Potential Ihres IBM-PC · Einführung in IBM BASIC · Überblick über die Hardware des IBM-PC · Erklärung der Textmodusgrafik, Farbgrafik und des Zeichentricks anhand vollständiger Programmbeispiele · IBM DOS erforderlich.
**Best.-Nr. MT 707, DM 48,— (Sfr. 44,20/öS 374,40)**

J. J. Parker
**Kommerzielle Anwendungen in Basic für den IBM-PC**
**August 1984, 370 Seiten**
Einführung in die Basic-Programmierung · Erstellung von Dateien · der Gebrauch von Listen und Tabellen · Verwendung von Direktzugriffs und sequentiellen Dateien · Gebrauch und Entwurf komplexer Programme · Einführung in Visicalc ·
**Best.-Nr. MT 719, DM 56,— (Sfr. 51,50/öS 436,80)**

J. E. Kelley jr.
**Professionelle Standardsoftware auf dem IBM-PC**
**Juli 1984, 587 Seiten**
Ein Arbeitsbuch für jeden IBM-PC-Benutzer · visuelles Programmieren mit WordStar, VisiCalc und dBase II · alles über Formatierung und die Erstellung von Sicherungsdisketten · Vorbereitung der programmspezifischen Disketten · mit Beispieldiskette im Buch.
**Best.-Nr. MT 709, DM 98,— (Sfr. 90,20/öS 764,40)**

D.P. Curtin/J.R. Alves/A.K. Briggs
**Mehr Gewinn durch: Erfolgskontrolle**
**am Beispiel IBM-PC & XT mit VisiCalc Advanced**
**1984, 198 Seiten**
Business-Fragen durch einfache Wirtschaftlichkeitsberechnungen beantwortet · Gewinn- und Verlustrechnung · Bilanz · Verhältniszahlen · für Unternehmensführer der vielversprechende Leitfaden zum Erfolg.
**Best.-Nr. MT 639, DM 42,— (Sfr. 38,60/öS 327,60)**

J. R. Alves/D. P. Curtin/A. K. Briggs
**Mehr Gewinne durch: Planung und Budgetierung**
**am Beispiel IBM-PC & XT mit VisiCalc Advanced**
**September 1984, 204 Seiten**
Planung und Budgetierung — eine Einführung · das Mittelfluß-Budget · Plan-Gewinn- und Verlustrechnung · Plan-Bilanz · Analyse durch Was-Wenn-Fragen · Anpassung des Beispiels an Ihr Unternehmen.
**Best.-Nr. MT 640, DM 42,— (Sfr. 38,60/öS 327,60)**

W. R. Osgood/J. F. Molloy jr.
**Mehr Gewinn durch: Gezielte Entscheidungen am**
**Beispiel IBM-PC & XT mit VisiCalc Advanced**
**Oktober 1984, 220 Seiten**
Ein umfassender Leitfaden für den Einsatz des IBM-PC und VisiCalc im Unternehmen · Preisbildung · Rabatte · Gewinn-, Verlustrechnung · für den erfolgsorientierten Unternehmer eine unentbehrliche Hilfe bei den täglichen Entscheidungen!
**Best.-Nr. MT 641, DM 42,— (Sfr. 38,60/öS 327,60)**

L. A. Wortman
**Business-Lösungen mit dem IBM-PC & XT**
**1984, 507 Seiten**
Alles über die Einsatzmöglichkeiten des IBM-PC in der Praxis · Auswahl und Anwendung von Sprachen und Dialekten · detaillierte Beschreibung aller Befehle · für den erfolgsorientierten Anwender.
**Best.-Nr. MT 689, DM 58,— (Sfr. 53,40/öS 452,40)**

N. Graham: **Programmieren mit dem IBM-PC: Basic**
**1984, 442 Seiten**
Basic-Grundlagen für den IBM-PC · Datentypen, Konstanten, Berechnungen und Ausgabe · Variablen: Zuordnung und Input · Programmentwurf · Fehlersuche · Strings · Sequentielle Dateien · Direktzugriffsdateien · Farbe und Grafik.
**Best.-Nr. MT 663, DM 58,— (Sfr. 53,40/öS 452,40)**

Die angegebenen Preise sind Ladenpreise

## Sie erhalten Markt & Technik-Bücher bei Ihrem Buchhändler

**Markt & Technik Verlag Aktiengesellschaft Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar**

# Weitere Fachbücher aus unserem Verlagsprogramm

W. B. Sanders: **Einführungskurs: Apple**
**Juli 1984, 297 Seiten**
Ein Begleitbuch für die ersten Schritte auf dem Apple II+ — Computer in der Programmiersprache Basic · logisch aufgebaute Kapitel · Vorschläge für Dienstprogramme · Programmbeschreibungen für kommerzielle Anwendungen und zur Textverarbeitung
**Best.-Nr. MT 745, DM 38,— (Sfr. 35,—/öS 296,40)**

M. K. Miller/M. A. Myers: **Macintosh**
**August 1984, 159 Seiten**
Alles über Macintosh — das neue Personal Computer-Konzept von Apple · Beschreibung des elektronischen Schreibtischs mit dazugehörigen Utensilien · Funktion und Bedienung der Maus · die Fenstertechnik · Grundlagen der Textverarbeitung · Glossar · für die richtige Kaufentscheidung ·
**Best.-Nr. MT 629, DM 44,— (Sfr. 40,50/öS 343,20)**

M. J. Capella/M. D. Weinstock: **Spiele für den Apple**
**Juli 1984, 270 Seiten**
Eine Sammlung von bewährten alten und raffinierten neuen Spielen für Ihren Apple-Computer · Programmierung in Applesoft-Basic · mit leicht verständlichen Einleitungen, die Ihnen den Spielablauf und die Programmiertricks erklären · das Spielebuch mit Lerneffekt.
**Best.-Nr. MT 725, DM 38,— (Sfr. 35,—/öS 296,40)**

M. J. Winter: **Lehrspielzeug Computer: Apple**
**Juli 1984, 139 Seiten**
Ein Buch für Kinder ab 8 Jahren, die Spaß haben an Worten, Zahlen und Bildern auf dem Apple II, II+, IIe · gelernt werden die wichtigsten Basic-Befehle, die Erstellung von Spielprogrammen und Grafiken · auch zur Vertiefung der Rechenkenntnisse geeignet ·
**Best.-Nr. MT 694, DM 24,80 (Sfr. 23,—/öS 193,40)**

W. B. Sanders: **Einführungskurs: IBM-PC/PC jr.**
**August 1984, 276 Seiten**
Ein neuer Einführungskurs des Erfolgsautors Sanders · was Sie über den Umgang mit Ihrem IBM-PC wissen müssen · schrittweise Einführung in IBM Microsoft Basic · Erstellen eines Programms · mit vielen Programmhilfen und nützlichen Hinweisen · ein ausgezeichnetes Anfänger-Buch, mit dem Sie experimentieren können!
**Best.-Nr. 726, DM 49,— (Sfr. 45,10/öS 382,20)**

De Voney/Lobb: **Der IBM-PC junior**
**1984, 286 Seiten**
Alles über die Fähigkeiten des neuen Computers · Hardwarekomponenten · Softwarepakete · Beschreibung des Plattenbetriebssystems (PC DOS 2.1), der Basic-Version für Magnetbandkassetten und der Floppy-Disk · Einsatzmöglichkeiten zur Steigerung der persönlichen Produktivität, Schulung, Unterhaltung.
**Best.-Nr. MT 660, DM 48,— (Sfr. 44,20/öS 374,40)**

Ed. Faulk: **Programmiertechniken auf dem IBM-PC**
**Oktober 1984, 331 Seiten**
Eine Einführung in die Methodik der Software-Entwicklung · die modernen Techniken der modularen und strukturierten Programmierung · Test und Fehlerbehebung für Basic-Programmierer · sowohl für »Neulinge« als auch für Fortgeschrittene.
**Best.-Nr. MT 721, DM 48,— (Sfr. 44,20/öS 374,40)**

R. Ashley/J.N. Fernandez
**PC-DOS: Das Betriebssystem des IBM-PC**
**1984, 381 Seiten**
Einführung in die Betriebssysteme DOS 1.1 und DOS 2.0 · Programme für Routinearbeiten · Programme starten · Dateien kreieren · löschen oder kopieren · Daten aus Dateien drucken · Platteninhaltsverzeichnisse auflisten.
**Best.-Nr. MT 643, DM 58,— (Sfr. 53,40/öS 452,40)**

C. Townsend: **MS-DOS**
**1984, 150 Seiten**
Alle Leistungsmerkmale von MS-DOS · Diskussion der Systembefehle anhand einfacher Beispiele · Einsatzmöglichkeiten höherer Programmsprachen · Befehlssatz in Tabellenform · systemspezifische Fehlermeldungen · Vergleich zwischen CP/M-86 und MS-DOS.
**Best.-Nr. MT 616, DM 43,— (Sfr. 39,60/öS 335,40)**

J. R. Groff/P. N. Weinberg: **Einführung in Unix**
**1984, 298 Seiten**
Ein umfassender Einblick in das Betriebssystem von Unix · das Dateisystem · Shell · Mehrbenutzerbetrieb · Textverarbeitung und Bürounterstützung · Softwareentwicklung · Fernverarbeitung · Markttrends · für jeden, der bereits vertraut ist mit Computersystemen · für Seminare oder Kurse.
**Best.-Nr. MT 688, DM 58,— (Sfr. 53,40/öS 452,40)**

J. N. P. Hume/R. C. Holt: **Pascal unter Unix**
**Juli 1984, 505 Seiten**
Eine Einführung in die Strukturierte Programmierung · ausführliche Beschreibung der Grundkonzepte des Software-Engineering · Einsatzmöglichkeiten in den Bereichen Handel, Wirtschaft, Recht und Sozialwissenschaften · geeignet auch für den Nicht-Mathematiker, sei er EDV-Anfänger oder Profi.
**Best.-Nr. MT 683, DM 58,— (Sfr. 53,40/öS 452,40)**

P. M. Chirlian: **Microsoft Fortran**
**August 1984, 451 Seiten**
Eine besonders für Anfänger geeignete gründliche Einführung in Microsoft Fortran · die Techniken des strukturierten Programmierens · die Erstellung von Programmen mit Beispielen für den TRS-80 und andere Rechner · Fehlerbehandlung · der Microsoft Editor · mit ausführlichem Glossar im Anhang.
**Best.-Nr. MT 717, DM 56,— (Sfr. 51,50/öS 436,80)**

W. Maaß: **Software-Schnellkurs: MS-DOS**
**1984, 88 Seiten**
MS-DOS für den Alltag · die Handhabung von Dateien bei Disketten und Festplatten · für DOS 2.0 als MS- und PC-Version.
**Best.-Nr. MT 651, DM 37,— (Sfr. 34,—/öS 288,60)**

P. Eipper: **COBOL-Handbuch für Microcomputer**
**September 1984, 297 Seiten**
Die COBOL-Befehlsstruktur · Bildschirmsteuerung · Dateiverwaltung · Compiler-Handhabung: ein syntaktischer Vergleich (COBOL-80, MS-COBOL, CIS-COBOL, RM-COBOL, LEVEL 2 COBOL, MBP-COBOL) · für Anfänger ein leicht verständlicher Einstieg in COBOL, für Profis ein übersichtliches Nachschlagewerk
**Best.-Nr. MT 747, DM 52,— (Sfr. 47,80/öS 405,60)**

Die angegebenen Preise sind Ladenpreise

## Sie erhalten Markt & Technik-Bücher bei Ihrem Buchhändler

* Sollten Sie diese Bücher ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
**Markt & Technik Verlag Aktiengesellschaft Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220**

# Weitere Fachbücher aus unserem Verlagsprogramm

J. D. Dennon: **CP/M Anatomie eines Betriebssystems**
**Dezember 1984, 321 Seiten**
Nützliche Utilities und Informationen, die nicht im Handbuch stehen · Handhabung des Debuggers und Assemblers · Aufbau und Funktionsweise des CCP, des Directory Ihrer Diskette und der Systemkomponenten BDOS und BIOS · für echte CP/M-Freaks eine Fundgrube an Insider-Tips!
**Best.-Nr. MT 763, DM 68,— (Sfr. 62,60/öS 530,40)**

P. Lüke: **iAPX 186 — Der Superchip**
**1984, 237 Seiten**
iAPX 186, die neueste Zentraleinheit von Intel · eine CPU-Familie mit einer Datenbreite von 16 Bit · softwarekompatibel zur 8086-Familie · mit Schaltungsbeispielen für den mühelosen Aufbau leistungsstarker Rechnerstrukturen · ein Leitfaden für den ambitionierten Computer-Hobbyisten und den Fachmann.
**Best.-Nr. MT 698, DM 42,— (Sfr. 38,60/öS 327,60)**

D. P. Curtin
**Handbuch der Textverarbeitung: Wordstar deutsch**
**1984, 312 Seiten**
Eine leicht verständliche Programmierhilfe · Aufbau und Formatierung der Bildschirmanzeige · Texteingabe und Textbearbeitung · die Arbeit mit Textblöcken · Formatieren zum Druck · Mail Merge · interessant auch für den erfahrenen Anwender.
**Best.-Nr. MT 686, DM 54,— (Sfr. 49,70/öS 421,20)**

W. Maaß: **Software-Praxis: Den Umgang mit WordStar schnell und einfach lernen**
**Band 1: Buch für Beginner**
**August 1984, 147 Seiten**
Für IBM-PC & XT und 100 %-IBM-kompatible Computer · Vorstellung der umfangreichen Menüs · Eingabe von Texten, Korrektur, Formatierung · Steuern verschiedener Schrift- und Druckformen u.v.a.m. Inkl. Probierdiskette und vielen Beispielen.
**Best.-Nr. 761, DM 88,— (Sfr. 81,—/öS 686,40)**

W. Maaß: **Software-Praxis: Den Umgang mit WordStar schnell und einfach lernen**
**Band 2: Buch für Fortgeschrittene**
**August 1984, 150 Seiten**
Gestalten von Kopf- und Fußzeilen · Suchen und ersetzen einzelner Worte oder Passagen · Verschieben von Textpassagen innerhalb einer Datei und zwischen verschiedenen Dateien · Steuerzeichen für den Ausdruck. Mit allen Beispielen auf Diskette.
**Best.-Nr. MT 762, DM 74,— (Sfr. 68,10/öS 577,20)**

W. Maaß: **Software-Schnellkurs: WordStar**
**1984, 88 Seiten**
Was man für den Umgang mit WordStar wissen muß · alles Wissenswerte in Kurzform · WordStar kurz und knapp erklärt.
**Best.-Nr. MT 609, DM 37,— (Sfr. 34,—/öS 288,60)**

W. Maaß: **Software-Praxis. Den Umgang mit Mail-Merge schnell und einfach lernen**
**Band 1: Buch für Beginner**
**August 1984, 153 Seiten**
Für IBM-PC & XT und 100 %-IBM-kompatible Computer · Erstellung von Datendateien · Erstellen von Standard-Briefen und Texten · Daten aus Adreßdaten nutzbar machen · Aufnehmen zusätzlicher Daten. Inkl. Probierdiskette und vielen Beispielen (PC/MS-DOS).
**Best.-Nr. MT 764, DM 88,— (Sfr. 81,—/öS 686,40)**

W. Maaß: **Software-Praxis. Den Umgang mit Mail-Merge schnell und einfach lernen**
**Band 2: Buch für Fortgeschrittene**
**August 1984, 150 Seiten**
Abfrage von Daten für Serienbriefe · Zusatzinformationen über den Bildschirm · Zusammenstellung von Texten aus Textbausteinen · Programmieren individueller Formulare. Alle Beispiele auf Diskette (PC/MS-DOS).
**Best.-Nr. MT 765, DM 74,— (Sfr. 68,10/öS 577,20)**

W. Bartel: **Textverarbeitung von Microsoft: WORD deutsche Version. Oktober 1984, 166 Seiten**
**Oktober 1984, 166 Seiten**
Ein ausführliche Anleitung mit Beispielen zum Selbststudium aller wichtigen Funktionen in WORD · Erstellung und Benutzung von Druckformatvorlagen · Erstellung von Mehrspaltentext · Serientext · geeignet für jeden Computerprofi.
**Best.-Nr. MT 802, DM 48,— (Sfr. 44,20/öS 374,40)**

Dr. P. Albrecht: **Das Datenbanksystem dBase II**
**1983, 280 Seiten**
Eine übersichtliche Anleitung für den praktischen Umgang mit dBaseII · Dateistrukturen schnell und einfach definiert, benutzt und geändert · hohe Flexibilität im Datenzugriff ohne starre Zugriffspfade · integrierte Kommandosprache, die eine komplette Anwendungsprogrammierung zuläßt · auch für Mikrocomputer-Neulinge.
**Best.-Nr. MT 524, DM 68,— (Sfr. 62,60/öS 530,40)**

Robert A. Byres:
**Einführung in Datenbanksysteme mit dBase II**
**1983, 280 Seiten**
Der leichtverständliche Einstieg in Datenbanken und ihrer Arbeitsweise · Grundlagen · Planung · Aufbau · Änderung · Pflege · Verwaltung und Anwendung von Datenbanken.
**Best.-Nr. MT 526 (Buch)**
**DM 68,— (Sfr. 62,80/öS 530,40)**
**Best.-Nr. MT 622 (Beispiele auf Diskette)**
**DM 48,— (Sfr. 48,—/öS 432,—)**

A. B. Green: **dBase II richtig eingesetzt**
**1983, 229 Seiten**
Grundlagen von dBase II (PC/MS-DOS) · Einführung in die Programmierung · Standardroutinen · Entwurf eines kompletten dBase-Systems · Datenaustausch mit Multiplan und Wordstar.
**Best.-Nr. MT 541, DM 68,— (Sfr. 62,60/öS 530,40)**
**Best.-Nr. MT 544 (Disk.), DM 48,— (Sfr. 48,—/öS 432,—)**

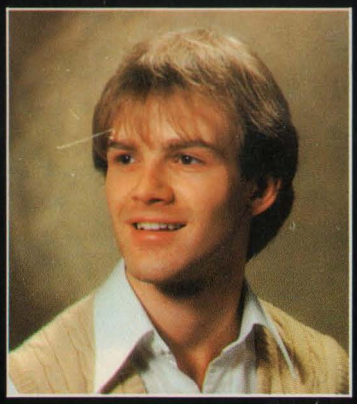
W. Maaß: **Software-Schnellkurs: dBase II**
**1984, 110 Seiten**
Das Datenbanksystem für Mikrocomputer kurz und bündig erklärt · eine praktische Kurzbeschreibung für Eilige.
**Best.-Nr. MT 607, DM 37,— (Sfr. 34,—/öS 288,60)**

W. Maaß: **Software-Praxis. Den Umgang mit dBase II schnell und einfach lernen.**
**Bd. 1: Buch für Laien**
**August 1984, 144 Seiten**
Datenbanken erstellen, Daten eingeben, korrigieren, sortieren, drucken. Datensätze suchen. Mit Probierdiskette und vielen Beispielen (PC/MS-DOS).
**Best.-Nr. MT 674, DM 88,— (Sfr. 81,—/öS 686,40)**

Die angegebenen Preise sind Ladenpreise

## Sie erhalten Markt & Technik-Bücher bei Ihrem Buchhändler

**Markt & Technik Verlag Aktiengesellschaft, Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar**



FRANZ SANTJOHANSER geboren 1961, beschäftigt sich seit 1978 mit der Programmierung von Microcomputern. Durch seine Tätigkeit bei der Entwicklung von Hardwarebaugruppen für den Apple II und andere 6502-Rechner, konnte er genügend Erfahrung sammeln, die ihm bei seiner Autorenarbeit nützlich sind. Nach einer Ausbildung und Beschäftigung bei der Firma Siemens AG, Zentrallabor für Kommunikationstechnik, und zweijähriger Bundeswehrzeit, kam F. Santjohanser Anfang 1984 zu Markt & Technik Verlag AG. Er ist dort im Buchverlag im technischen Lektorat tätig.

# Das Apple IIc Buch

Das große Apple IIc-Buch will Ihnen all das nahe bringen, was andere Dokumentationen so oft verschweigen. Es umfaßt eine umfangreiche und gegliederte Hardwarebesprechung, sowie eine ausführliche Einarbeitung in Applesoft-BASIC, Integer-BASIC und Maschinensprache. In kleinen Schritten werden Sie an ein Ziel herangeführt, daß es Ihnen schnell ermöglicht, eigene wirkungsvolle Programme zu schreiben. Anhand kurzer praktischer Beispiele wird die Funktion und Wirkungsweise bestimmter Befehle gezeigt. Am Ende jedes Kapitels werden Sie zur Durcharbeitung einiger kurzer Fragen angehalten, um so das gelernte noch einmal zu wiederholen.

Sie durchwandern eine Welt der Bits und Bytes und werden schon nach wenigen Lektionen Ihren Apple IIc besser kennenlernen und Spaß haben mit ihm zu arbeiten.

Umfangreiche Tabellen und Diagramme zeigen Ihnen in verständlicher Weise die Arbeitsabläufe, die in Ihrem Computer vorgehen. Sie erlernen also nicht nur die Bedienung Ihres Apple IIc, sondern werden auch die Zusammenhänge besser verstehen.

Mit Hilfe von Maschinenprogrammen dringen Sie immer tiefer in die Struktur Ihres Rechners ein und können so seine Möglichkeiten voll ausschöpfen. Auch das Programmieren der »Maus«, die angeschlossen werden kann, wird ausführlich besprochen. Weiter die Grafikprogrammierung (auch 3-D), Tonerzeugung und Animation.

Viele der gezielten Tips und Tricks werden selbst für Apple IIplus, IIe-Anwender neu und interessant sein. Die Betriebssysteme DOS 3.3 und ProDOS werden miteinander verglichen und in praktischen Beispielen erklärt. So erhalten Sie mit diesem Buch eine umfangreiche Ergänzung zu Ihrer Apple IIc-Dokumentation und ein praktisches Nachschlagewerk für Ihre Programmierprobleme.

Folgende Hardware wird zur Durcharbeitung empfohlen:
— Apple IIc und Monitor oder Fernsehgerät,
— Apple Maus und Joystick,
— evtl. externes Diskettenlaufwerk.

Folgende Software wird zur Durcharbeitung empfohlen:
— System-Dienstprogramme Apple IIc,
— Spaß mit Apple.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-89090-078-X

DM 52,—
Sfr. 47,80 / öS 405,60